Vendor: Oracle

Exam Code: 1Z0-805

Exam Name: Upgrade to Java SE 7 Programmer

Version: Demo

QUESTION NO: 1

Which statement is true about the take method defined in the WatchService interface?

A. Retrieves and removes the next watch key, or returns null of none are present.
B. Retrieves and removes the next watch key. If a queued key is not immediately available, the program waits for the specified wait time.
C. Retrieves and removes the next watch key: waits if no key is yet present.
D. Retrieves and removes all pending events for the watch key, returning a list of the events that were retrieved.

Answer: C
Explanation: The WatchKey take() method retrieves and removes next watch key, waiting if none are yet present.

Note: A watch service that watches registered objects for changes and events. For example a file manager may use a watch service to monitor a directory for changes so that it can update its display of the list of files when files are created or deleted.
A Watchable object is registered with a watch service by invoking its register method, returning a WatchKey to represent the registration. When an event for an object is detected the key is signalled, and if not currently signalled, it is queued to the watch service so that it can be retrieved by consumers that invoke the poll or take methods to retrieve keys and process events. Once the events have been processed the consumer invokes the key's reset method to reset the key which allows the key to be signalled and re-queued with further events.

Reference: Interface WatchService

QUESTION NO: 2

Given the code fragment:

```
private static void copyContents (File source, File target) {

try {inputStream fis = new FileInputStream(source);

outputStream fos = new FileOutputStream (target);

byte [] buf = new byte [8192];
```

```
int i;

while ((i = fis.read(buf)) != -1) {

fos.write (buf, 0, i);

}

 //insert code fragment here. Line **

System.out.println ("Successfully copied");

}
```

Which code fragments, when inserted independently at line **, enable the code to compile?

A. }catch (IOException | NoSuchFileException e)
{ System.out.println(e);
}
B. } catch (IOException | IndexOutOfBoundException e)
{ System.out.println(e);
}
C. } catch (Exception | IOException | FileNotFoundException e)
{ System.out.println(e);
}
D. } catch (NoSuchFileException e)
{ System.out.println(e);
}
E. } catch (InvalidPathException | IOException e)
{ System.out.println(e);
}

Answer: B,D,E
Explanation: B: Two mutually exclusive exceptions. Will work fine.
D: A single exception. Will work fine.
E: Two mutually exclusive exceptions. Will work fine.

Note: In Java SE 7 and later, a single catch block can handle more than one type of exception. This feature can reduce code duplication and lessen the temptation to catch an overly broad exception.
In the catch clause, specify the types of exceptions that block can handle, and separate each exception type with a vertical bar (|).

Note 2:NoSuchFileException: Checked exception thrown when an attempt is made to access a file

that does not exist.
InvalidPathException: Unchecked exception thrown when path string cannot be converted into a Path because the path string contains invalid characters, or the path string is invalid for other file system specific reasons.
FileNotFoundException: Signals that an attempt to open the file denoted by a specified pathname has failed.
This exception will be thrown by the FileInputStream, FileOutputStream, and RandomAccessFile constructors when a file with the specified pathname does not exist. It will also be thrown by these constructors if the file does exist but for some reason is inaccessible, for example when an attempt is made to open a read-only file for writing.

QUESTION NO: 3

Which two statements are true about the walkFileTree method of the files class?

A. The file tree traversal is breadth-first with the given FileVisitor invoked for each file encountered.
B. If the file is a directory, and if that directory could not be opened, the postVisitFileFailed method is invoked with the I/O exception.
C. The maxDepth parameter's value is the maximum number of directories to visit.
D. By default, symbolic links are not automatically followed by the method.

Answer: C,D
Explanation: C: The method walkFileTree(Path start, Set<FileVisitOption> options, int maxDepth, FileVisitor<? super Path> visitor)
walks a file tree.
The maxDepth parameter is the maximum number of levels of directories to visit. A value of 0 means that only the starting file is visited, unless denied by the security manager. A value of MAX_VALUE may be used to indicate that all levels should be visited. The visitFile method is invoked for all files, including directories, encountered at maxDepth, unless the basic file attributes cannot be read, in which case the visitFileFailed method is invoked.

D:You need to decide whether you want symbolic links to be followed. If you are deleting files, for example, following symbolic links might not be advisable. If you are copying a file tree, you might want to allow it. By default, walkFileTree does not follow symbolic links.

Reference: The Java Tutorials, Walking the File Tree

Reference: walkFileTree

QUESTION NO: 4

Which code fragments print 1?

A. String arr [] = {"1", "2", "3"};
List <? extendsString > arrList = new LinkedList <> (Arrays.asList (arr));
System.out.println (arrList.get (0));
B. String arr [] = {"1", "2", "3"};
List <Integer> arrList = new LinkedList <> (Arrays.asList (arr));
System.out.println (arrList.get (0));
C. String arr [] = {"1", "2", "3"};
List <?> arrList = new LinkedList <> (Arrays.asList (arr));
System.out.println (arrList.get (0));
D. String arr [] = {"1","2","3"};
List <?> arrList = new LinkedList <?>(Arrays.asList (arr));
System.out.println (arrList.get (0));
E. String arr [] = {"1","2","3"};
List <Integer> extendsString > arrList =new LinkedList <Integer> (Arrays.asList (arr));
System.out.println (arrList.get (0));

Answer: A,C
Explanation:

Note:You can replace the type arguments required to invoke the constructor of a generic class with an empty set of type parameters (<>) as long as the compiler can infer the type arguments from the context. This pair of angle brackets is informally called the diamond.

QUESTION NO: 5

Given the code fragment:

public static void main(String[] args)

{ String source =

"d:\\company\\info.txt";

String dest = "d:\\company\\emp\\info.txt";

//insert code fragment here Line **

} catch (IOException e) {

System.err.println ("Caught IOException: " + e.getmessage();

}

}

Which two try statements, when inserted at line **, enable the code to successfully move the file info.txt to the destination directory, even if a file by the same name already exists in the destination directory?

A. try {FileChannel in = new FileInputStream(source).getChannel();
FileChannel out = new FileOutputStream(dest).getChannel ();
in.transferTo (0, in.size(), out);
B. try {Files.copy(Paths.get(source), Paths.get(dest));
Files.delete(Paths.get(source));
C. try {Files.copy(Paths.get(source), Paths.get(dest));
Files.delete(Paths.get(source));
D. try {Files.move(Paths.get(source),Paths.get(dest));
E. try {BufferedReader br = Files.newBufferedReader(Paths.get(source), Charset.forName ("UTF-8"));
BufferedWriter bw = Files.newBufferedWriter (Paths.get(dest), Charset.forName ("UTF-8"));
String record = "";
while ((record = br.readLine()) != null){
bw.write (record);
bw.newLine();
}
Files.delete(Paths.get(source));

Answer: B,D
Explanation:

QUESTION NO: 6

What design pattern does the Drivermanager.getconnection () method characterize?

A. DAO
B. Factory
C. Singleton
D. composition

Answer: B
Explanation: DriverManager has a factory method getConnection() that returns a Connection object.

Note 1:A factory method is a method that creates and returns new objects.
The factory pattern (also known as the factory method pattern) is a creational design pattern. A

factory is a Java class that is used to encapsulate object creation code. A factory class instantiates and returns a particular type of object based on data passed to the factory. The different types of objects that are returned from a factory typically are subclasses of a common parent class.

Note 2:
The method DriverManager.getConnection establishes a database connection. This method requires a database URL, which varies depending on your DBMS. The following are some examples of database URLs: MySQL, Java DB.

QUESTION NO: 7

Given the code fragment:

dataFormat df;

Which statement defines a new DataFormat object that displays the default date format for the UK Locale?

A. df = DateFormat.getDateInstance (DateFormat.DEFAULT, Locale(UK));
B. df = DateFormat.getDateInstance (DateFormat.DEFAULT, UK);
C. df = DateFormat.getDateInstance (DateFormat.DEFAULT, Locale.UK);
D. df = new DateFormat.getDataInstance (DataFormat.DEFAULT, Locale.UK);
E. df = new DateFormat.getDateInstance (DateFormat.DEFAULT, Locale(UK));

Answer: C
Explanation: DateFormat is an abstract class that provides the ability to format and parse dates and times. The getDateInstance() method returns an instance of DateFormat that can format date information. It is available in these forms:
static final DateFormat getDateInstance( )
static final DateFormat getDateInstance(int style)
static final DateFormat getDateInstance(int style, Locale locale)

The argument style is one of the following values: DEFAULT, SHORT, MEDIUM, LONG, or FU
These are int constants defined by DateFormat.

QUESTION NO: 8
Given three resource bundles with these values set for menu1: (The default resource bundle is

English US resource Bundle

Menu1 = small

French resource Bundle

Menu1 = petit

Chinese Resource Bundle

Menu = 1

And given the code fragment:

Locale.setDefault (new Locale("es", "ES")); // Set default to Spanish and Spain

loc1 = Locale.getDefault();

ResourceBundle messages = ResourceBundle.getBundle ("messageBundle", loc1);

System.out.println (messages.getString("menu1"));

What is the result?

A. No message is printed
B. petit
C. :
D. Small
E. A runtime error is produced

Answer: E
Explanation: Compiles fine, but runtime error when trying to access the Spanish Resource bundle (which does not exist):
Exception in thread "main" java.util.MissingResourceException: Can't find bundle for base name messageBundle, locale es_ES

QUESTION NO: 9

```
import java.util.*;
public class StringApp {
public static void main (String [] args)
{ Set <String> set = new TreeSet <> ();
set.add("X");
set.add("Y");
set.add("X");
set.add("Y");
set.add("X");
Iterator <String> it = set.iterator ();
int count = 0;
while (it.hasNext())
{ switch
(it.next()){ case "X":
System.out.print("X  ");
break; case "Y":
System.out.print("Y  ");
break;
}
count++;
}
System.out.println ("\ncount = " + count);
}
}
```

A. X X Y X Y
count = 5
B. X Y X Y
count = 4
C. X Y
count = s
D. X Y
count = 2

Answer: D
Explanation: A set is a collection that contains no duplicate elements. So set will include only two elements at the start of while loop. The while loop will execute once for each element. Each element will be printed.

Note:
*public interface Iterator
An iterator over a collection. Iterator takes the place of Enumeration in the Java collections framework. Iterators differ from enumerations in two ways:
Iterators allow the caller to remove elements from the underlying collection during the iteration with well-defined semantics.
Method names have been improved.
*hasNext
public boolean hasNext()
Returns true if the iteration has more elements. (In other words, returns true if next would return element rather than throwing an exception.)
*next
publicObjectnext()
Returns the next element in the iteration.

QUESTION NO: 10

Given the code fragment:

List<Person> pList = new CopyOnWriteArrayList<Person>();

Which statement is true?

A. Read access to the List should be synchronized.
B. Write access to the List should be synchronized.
C. Person objects retrieved from the List are thread-safe.
D. A Person object retrieved from the List is copied when written to.
E. Multiple threads can safely delete Person objects from the List.

Answer: C
Explanation: CopyOnWriteArrayList produces a thread-safe variant of ArrayList in which all mutative operations (add, set, and so on) are implemented by making a fresh copy of the underlying array.

Note: his is ordinarily too costly, but may be more efficient than alternatives when traversal operations vastly outnumber mutations, and is useful when you cannot or don't want to synchronize traversals, yet need to preclude interference among concurrent threads. The "snapshot" style iterator method uses a reference to the state of the array at the point that the iterator was created. This array never changes during the lifetime of the iterator, so interference is impossible and the iterator is guaranteed not to throw ConcurrentModificationException. The iterator will not reflect additions, removals, or changes to the list since the iterator was created. Element-changing operations on iterators themselves (remove, set, and add) are not supported. These methods throw UnsupportedOperationException.

All elements are permitted, including null.

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a CopyOnWriteArrayList happen-before actions subsequent to the access or removal of that element from the CopyOnWriteArrayList in another thread.

Reference: java.util.concurrent.CopyOnWriteArrayList<E>

QUESTION NO: 11

Given the fragment:

```
public class CustomerApplication {

public static void main (String [] args) {

CustomerDAO custDao = new CustomerDAOMemoryImp1 ();

// . . . other methods

}
```

}

Which two valid alternatives to line 3 would decouple this application from a specific implementation of customerDAO?

A. CustomerDAO custDao = new customerDAO();
B. CustomerDAO custDao = (CustomerDAO) new object();
C. CustomerDAO custDao = CustomerDAO.getInstance();
D. CustomerDAO custDao = (CustomerDAO) new CustomerDAOmemoryImp1();
E. CustomerDAO custDao = CustomerDAOFactory.getInstance();

Answer: C,E
Explanation:
Note: Data Access Layer has proven good in separate business logic layer and persistent layer. The DAO design pattern completely hides the data access implementation from its clients. The interfaces given to client does not changes when the underlying data source mechanism changes. this is the capability which allows the DAO to adopt different access scheme without affecting to business logic or its clients. generally it acts as a adapter between its components and database. The DAO design pattern consists of some factory classes, DAO interfaces and some DAO classes to implement those interfaces.

QUESTION NO: 12

Given a resource bundle MessageBundle, what is the name of the default bundle file?

A. MessageBundle.profile
B. MessageBundle.xml
C. MessageBundle.java
D. MessageBundle.properties

Answer: D
Explanation: A properties file is a simple text file. You should always create a default properties file. The name of this file begins with the base name of your ResourceBundle and ends with the .properties suffix.

Reference: The Java Tutorials,Backing a ResourceBundle with Properties Files

QUESTION NO: 13
Given the code fragment:

```
public class Test {

public static void main (String [] args) {

Path path1 = Paths.get("D:\\sys\\asm\\.\\data\\..\\..\\mfg\\production.log");

System.out.println(path1.normalize());

System.out.println(path1.getNameCount());

}

}
```

What is the result?

A. D:\sys\mfg\production.log
8
B. D:\\sys\\asm\\.\\data\\. . \\mfg\\production.log
6
C. D: \\sys\\asm\\.\\data\\. . \\mfg\\production.log
8
D. D: \sys\mfg\production.log
4
E. D: \\ sys\\asm\\data\\mfg\\production.log
6

Answer: A
Explanation: The normalize method removes any redundant elements, which includes any "." or "directory/.." occurrences.
ThegetNameCount method returns the number of elements in the path.Here there are 8 elements (in the redundant path).

Reference: The Java Tutorials,Path Operations

QUESTION NO: 14

You are using a database from XY/Data. What is a prerequisite for connecting to the database using a JDBC 4.0 driver from XY/Data?

A. Use the JDBC DriverManager.loadDriver method.

B. Put the XY/data driver into the classpath of your application.
C. Create an instance of the XY/Data driver class using the new keyword.
D. Create an Implementation of DriverManager that extends the XY/Data driver

Answer: B
Explanation: First, you need to establish a connection with the data source you want to use. A data source can be a DBMS, a legacy file system, or some other source of data with a corresponding JDBC driver. Typically, a JDBC application connects to a target data source using one of two classes:
* DriverManager: This fully implemented class connects an application to a data source, which is specified by a database URL. When this class first attempts to establish a connection, it automatically loads any JDBC 4.0 drivers found within the class path(B). Note that your application must manually load any JDBC drivers prior to version 4.0.
* DataSource: This interface is preferred over DriverManager because it allows details about the underlying data source to be transparent to your application. A DataSource object's properties are set so that it represents a particular data source.

Note:The JDBC Architecture mainly consists of two layers:
First is JDBC API, which provides the application-to-JDBC Manager connection.
Second is JDBC Driver API, which supports the JDBC Manager-to-Driver Connection. This has to provide by the vendor of database, you must have notice that one external jar file has to be there in class path for forth type of driver (B).
The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Reference: The Java Tutorials, Establishing a Connection

QUESTION NO: 15

Given the following code fragment:

```
public class Calc {

public static void main (String [] args) {

//* insert code here  Line **
```

System.out.print("The decimal value is" + var);

}

}

Which three code fragments, when inserted independently at line **, enable the code to compile/

A. int var = 0b_1001;
B. long var = 0b100_01L;
C. float var = 0b10_01; **D.**
float var = 0b10_01F; **E.**
double var = 0b10_01;
F. double var = 0b10_01D;

Answer: B,C,E
Explanation: B: output 17
C: output 9.0
E: output 9.0
Not A: A _ character cannot begin a number.
Not D: A float cannot be defined as a binary number (with literal B)
Not F: A float cannot be defined as a decimal number (with literal D)

Note1:
In Java SE 7 and later, any number of underscore characters (_) can appear anywhere between digits in a numerical literal. This feature enables you, for example. to separate groups of digits in numeric literals, which can improve the readability of your code.

For instance, if your code contains numbers with many digits, you can use an underscore character to separate digits in groups of three, similar to how you would use a punctuation mark like a comma, or a space, as a separator.

You can place underscores only between digits; you cannot place underscores in the following places:
* At the beginning or end of a number (not A)
* Adjacent to a decimal point in a floating point literal
* Prior to an F or L suffix
* In positions where a string of digits is expected

Note 2:An integer literal is of type long if it ends with the letter L or l; otherwise it is of type int. It is recommended that you use the upper case letter L because the lower case letter l is hard to distinguish from the digit 1.

Values of the integral types byte, short, int, and long can be created from int literals. Values of type long that exceed the range of int can be created from long literals. Integer literals can be expressed by these number systems:

Decimal: Base 10, whose digits consists of the numbers 0 through 9; this is the number system you use every day
Hexadecimal: Base 16, whose digits consist of the numbers 0 through 9 and the letters A through F
Binary: Base 2, whose digits consists of the numbers 0 and 1 (you can create binary literals in Java SE 7 and later)

Reference: The Java Tutorials,Primitive Data Types:

Using Underscore Characters in Numeric Literals

Integer Literals

QUESTION NO: 16

Given the code fragment:

```
String query = "SELECT ID FROM Employee";  \\ Line 1

try (Statement stmt = conn.CreateStatement()) {  \\ Line 2

ResultSet rs = stmt.executeQuery(query);  \\ Line 3

stmt.executeQuery ("SELECT ID FROM Customer");  \\ Line 4

while (rs.next()) {

\\process the results

System.out.println ("Employee ID: " + rs.getInt("ID") );

}

} catch (Exception e) {

system.out.println ("Error");

}
```

Assume that the SQL queries return records. What is the result of compiling and executing this code fragment?

A. The program prints employees IDs.
B. The program prints customer IDs.
C. The program prints Error.
D. Compilation fails on line 13.

Answer: A
Explanation: Line 3 sets the resultset rs. rs will contain IDs from the employee table.
Line 4 does not affect the resultset rs. It just returns a resultset (which is not used).
Note:
A ResultSet object is a table of data representing a database result set, which is usually generated by executing a statement that queries the database.
You access the data in a ResultSet object through a cursor. Note that this cursor is not a database cursor. This cursor is a pointer that points to one row of data in the ResultSet. Initially, the cursor is positioned before the first row. The method ResultSet.next moves the cursor to the next row. This method returns false if the cursor is positioned after the last row. This method repeatedly calls the ResultSet.next method with a while loop to iterate through all the data in the ResultSet.

Reference: The Java Tutorials,Retrieving and Modifying Values from Result Sets

QUESTION NO: 17

Given:

```
public class SampleClass {

public static void main(String[] args)

{ SampleClass sc = new

SampleClass(); sc.processCD();

}

private void processCD() {

try (CDStream cd = new CDStream()) {

cd.open();

cd.read();

cd.write("lullaby");
```

```
cd.close();
} catch (Exception e)
{ System.out.println("Exception
thrown");
}
}
class CDStream {
String cdContents = null;
public void open()
{ cdContents = "CD
Contents";
System.out.println("Opened CD stream");
}
public String read() throws Exception {
throw new Exception("read error");
}
public void write(String str)
{ System.out.println("CD str is: " +
str);
}
public void close() {
cdContents = null;
}
```

What is the result?

A. Compilation CD stream
B. Opened CD thrown
C. Exception thrown

D. Opened CD stream
CD str is: lullaby

Answer: A

Explanation: In this example the compilation of line " try (CDStream cd = new CDStream()) {" will fail, as
try-with-resources not applicable to variable type CDStream.

Note: The try-with-resources statement is a try statement that declares one or more resources. A resource is an object that must be closed after the program is finished with it. The try-with-resources statement ensures that each resource is closed at the end of the statement. Any object that implements java.lang.AutoCloseable, which includes all objects which implement java.io.Closeable, can be used as a resource.

Reference: The Java Tutorials,The try-with-resources Statement

QUESTION NO: 18

Two companies with similar robots have merged. You are asked to construct a new program that allows the features of the robots to be mixed and matched using composition. Given the code fragments:

```
public class CrusherRobot {

public void walk () {}

public void positionArm (int x, int y, int z) {}

public void raiseHammer() {}

public void dropHammer() {}

}

public class GripperRobot {

public void walk() {}

public void moveArm (int x, int y, int z) {}

public void openGripper () {}

public void closeGripper() {}

}
```

When applying composition to these two classes, what functionality should you extract into a new class?

A. A new BasicRobot class that provides walking.
B. A new BasicRobot class that combines gripping and hammering.
C. A new BasicRobotFactory class to construct instances of GripperRobot.
D. A new BasicRobotFactory class to construct instances of CrusherRobot.

Answer: B
Explanation:

QUESTION NO: 19

Which three must be used when using the Java.util.concurrent package to execute a task that returns a result without blocking?

A. ExecutorService
B. Runnable
C. Future
D. Callable
E. Thread
F. Executor

Answer: A,D,F
Explanation: The java.util.concurrent package defines three executor interfaces:
*(F)Executor, a simple interface that supports launching new tasks.
*(A)ExecutorService, a subinterface of Executor, which adds features that help manage the lifecycle, both of the individual tasks and of the executor itself.
* ScheduledExecutorService, a subinterface of ExecutorService, supports future and/or periodic execution of tasks.

Typically, variables that refer to executor objects are declared as one of these three interface types, not with an executor class type.

D: The ExecutorService interface supplements execute with a similar, but more versatile submit method. Like execute, submit accepts Runnable objects, but also accepts Callable objects, which allow the task to return a value.

Reference: The Java Tutorials,Executor Interfaces

QUESTION NO: 20

Which statement creates a low-overhead, low contention random number generator that is isolated to a thread to generate a random number between 1 and 100?

A. int i = ThreadLocalRandom.current().nextInt (1, 101);
B. int i = ThreadSaferandom.current().nextInt(1, 101);
C. int i = (int) Math.random()*.nextInt(1, 101);
D. int i = (int) Match.random (1, 101);
E. int i = new Random (). nextInt (100)+1;

Answer: A
Explanation: public class ThreadLocalRandom
extendsRandom
A random number generator isolated to the current thread. Like the global Random generator used by the Math class, a ThreadLocalRandom is initialized with an internally generated seed that may not otherwise be modified. When applicable, use ofThreadLocalRandom rather thanshared Random objects in concurrent programs will typically encounter much less overhead and contention. Use of ThreadLocalRandom is particularly appropriate when multiple tasks (for example, each a ForkJoinTask) use random numbers in parallel in thread pools.
Usages of this class should typically be of the
form: ThreadLocalRandom.current().nextX(...) (where X is Int, Long, etc). When all usages are c this form, it is never possible to accidently share a ThreadLocalRandom across multiple threads.
This class also provides additional commonly used bounded random generation methods.

QUESTION NO: 21

Given:

```
public class DataCache {

private static final DataCache instance = new DataCache ();

public static DataCache getInstance () {

return instance;

}
```

Which design pattern best describes the class?

A. Singleton
B. DAO
C. Abstract Factory
D. Composition

Answer: A
Explanation: Java has several design patterns Singleton Pattern being the most commonly used. Java Singleton pattern belongs to the family of design patterns, that govern the instantiation process. This design pattern proposes that at any time there can only be one instance of a singleton (object) created by the JVM.
The class's default constructor is made private, which prevents the direct instantiation of the object by others (Other Classes). A static modifier is applied to the instance method that returns the object as it then makes this method a class level method that can be accessed without creating an object.

QUESTION NO: 22

Given the code format:

SimpleDateFormat sdf;

Which code statements will display the full text month name?

A. sdf = new SimpleDateFormat ("mm", Locale.UK);
System.out.println ("Result: " + sdf.format(new Date()));
B. sdf = new SimpleDateFormat ("MM", Locale.UK);
System.out.println ("Result:"+ sdf.format(new Date()));
C. sdf = new SimpleDateFormat ("MMM", Locale.UK);
System.out.println ("Result:"+ sdf.format(new Date()));
D. sdf = new SimpleDateFormat ("MMMM", Locale.UK);
System.out.println ("Result:"+ sdf.format(new Date()));

Answer: D
Explanation: To getthe full length month name use SimpleDateFormat('MMMM').
Note:SimpleDateFormat is a concrete class for formatting and parsing dates in a locale-sensitive manner. It allows for formatting (date -> text), parsing (text -> date), and normalization.

SimpleDateFormat allows you to start by choosing any user-defined patterns for date-time formatting. However, you are encouraged to create a date-time formatter with either getTimeInstance, getDateInstance, orgetDateTimeInstance in DateFormat. Each of these class methods can return a date/time formatter initialized with a default format pattern. You may modify the format pattern using the applyPattern methods as desired.

QUESTION NO: 23

The default file system includes a logFiles directory that contains the following files:

log – Jan2009

log_01_2010

log_Feb2010

log_Feb2011

log-sum-2012

How many files the matcher in this fragment match?

PathMatcher matcher = FileSystems.getDefault ().getPathMatcher ("glob:???_*1");

A. One
B. Two
C. Three
D. Four
E. Five
F. Six

Answer: A
Explanation: The glob pattern is: any three characters, followed by _ ,followed by any number of characters, and ending with a 1.
Only log_Feb2011 matches this pattern.

Note:
You can use glob syntax to specify pattern-matching behavior.

A glob pattern is specified as a string and is matched against other strings, such as directory or file names. Glob syntax follows several simple rules:

* An asterisk, *, matches any number of characters (including none).
** Two asterisks, **, works like * but crosses directory boundaries. This syntax is generally used for matching complete paths.
* A question mark, ?, matches exactly one character.
* Braces specify a collection of subpatterns. For example:
{sun,moon,stars} matches "sun", "moon", or "stars."
{temp*,tmp*} matches all strings beginning with "temp" or "tmp."
* Square brackets convey a set of single characters or, when the hyphen character (-) is used, a range of characters. For example:
[aeiou] matches any lowercase vowel.
[0-9] matches any digit.
[A-Z] matches any uppercase letter.
[a-z,A-Z] matches any uppercase or lowercase letter.
* Within the square brackets, *, ?, and \ match themselves.
*All other characters match themselves.
* To match *, ?, or the other special characters, you can escape them by using the backslash character, \. For example: \\ matches a single backslash, and \? matches the question mark.

Reference: The Java Tutorials

Finding Files

What *Is* a Glob?

QUESTION NO: 24

Given the code fragment:

SimpleDateFormat sdf;

Which code fragment displays the two-digit month number?

A. sdf = new SimpleDateFormat ("mm", Locale.UK);
System.out.printIn (“Result: ” + sdf.format(new Date()))
B. sdf = new SimpleDateFormat ("MM", Locale.UK);
System.out.printIn (“Result: ” + sdf.format(new Date()))
C. sdf = new SimpleDateFormat ("MMM", Locale.UK);
System.out.printIn ("Result: "+ sdf.format(new Date()))
D. sdf = new SimpleDateFormat ("MMMM", Locale.UK);

System.out.println ("Result:"+ sdf.format(new Date()))

Answer: B
Explanation: B: Output example (displays current month numerically): 04

Note:SimpleDateFormat is a concrete class for formatting and parsing dates in a locale-sensitive manner. It allows for formatting (date -> text), parsing (text -> date), and normalization. SimpleDateFormat allows you to start by choosing any user-defined patterns for date-time formatting. However, you are encouraged to create a date-time formatter with either getTimeInstance, getDateInstance, orgetDateTimeInstance in DateFormat. Each of these class methods can return a date/time formatter initialized with a default format pattern. You may modify the format pattern using the applyPattern methods as desired.

QUESTION NO: 25

Which three enum constants are defined in FilevisitResult?

A. CONTINUE
B. SKIP_SIBLINGS
C. FOLLOW_LINKS
D. TERMINATE
E. NOFOLLOW_LINKS
F. DELETE_CHILD

Answer: A,B,D
Explanation: The FileVisitor methods return a FileVisitResult value. You can abort the file walking process or control whether a directory is visited by the values you return in the FileVisitor methods:

* CONTINUE – Indicates that the file walking should continue. If the preVisitDirectory method returns CONTINUE, the directory is visited.
* SKIP_SIBLINGS – When preVisitDirectory returns this value, the specified directory is not visited, postVisitDirectory is not invoked, and no further unvisited siblings are visited. If returned from the postVisitDirectory method, no further siblings are visited. Essentially, nothing further happens in the specified directory.
* TERMINATE – Immediately aborts the file walking. No further file walking methods are invoked after this value is returned.
* SKIP_SUBTREE – When preVisitDirectory returns this value, the specified directory and its subdirectories are skipped. This branch is "pruned out" of the tree.

Note:To walk a file tree, you first need to implement a FileVisitor. A FileVisitor specifies the required behavior at key points in the traversal process: when a file is visited, before a directory is

accessed, after a directory is accessed, or when a failure occurs.

Reference: The Java Tutorials,Walking the File Tree

QUESTION NO: 26

Given the code fragment:

```
public void processFile() throws IOException, ClassNotFoundException {
try (FileReader fr = new FileReader ("logfilesrc.txt");
FileWriter fw = new FileWriter ("logfiledest.txt"))
{ Class c = Class.forName ("java.lang.JString");
}
}
```

If exception occur when closing the FileWriter object and when retrieving the JString class object, which exception object is thrown to the caller of the processFile method?

A. java.io.IOException
B. java.lang.Exception
C. java.lang.ClassNotException
D. java.lang.NoSuchClassException

Answer: A
Explanation:

QUESTION NO: 27

Given the following incorrect program:

```
class MyTask extends RecursiveTask<Integer> {
final int low;
final int high;
static final int THRESHOLD =  /* . . . */
```

```
MyTask (int low, int high) { this.low = low; this.high = high; }

Integer computeDirectly()/* . . . */

protected void compute() {

if (high – low <= THRESHOLD)

return computeDirectly();

int mid = (low + high) / 2;

invokeAll(new MyTask(low,  mid), new MyTask(mid, high));
```

Which two changes make the program work correctly?

A. Results must be retrieved from the newly created MyTask Instances and combined.
B. The THRESHOLD value must be increased so that the overhead of task creation does not dominate the cost of computation.
C. The midpoint computation must be altered so that it splits the workload in an optimal manner.
D. The compute () method must be changed to return an Integer result.
E. The computeDirectly () method must be enhanced to fork () newly created tasks.
F. The MyTask class must be modified to extend RecursiveAction instead of RecursiveTask.

Answer: A,D
Explanation: D: the compute() method must return a result.
A: These results must be combined (in the lineinvokeAll(new MyTask(low, mid), new MyTask(mid, high));)

Note 1:A RecursiveTask is a recursive result-bearing ForkJoinTask.

Note 2: The invokeAll(ForkJoinTask<?>... tasks) forks the given tasks, returning when isDone holds for each task or an (unchecked) exception is encountered, in which case the exception is rethrown.

Note 3: Using the fork/join framework is simple. The first step is to write some code that performs a segment of the work. Your code should look similar to this:

```
if (my portion of the work is small enough)
do the work directly
else
split my work into two pieces
invoke the two pieces and wait for the results
```
Wrap this code as a ForkJoinTask subclass, typically as one of its more specialized types

RecursiveTask(which can return a result) or RecursiveAction.

QUESTION NO: 28

Given:

```
private static void copyContents() {

try (

InputStream fis = new FileInputStream("report1.txt");

OutputStream fos = new FileOutputStream("consolidate.txt");

) {

byte[] buf = new byte[8192];

int i;

while ((i = fis.read(buf)) != -1) {

fos.write(buf, 0, i);

}

fis.close();

fis = new FileInputStream("report2.txt");

while ((i = fis.read(buf)) != -1) {

fos.write(buf, 0, i);

}

}
```

What is the result?

A. Compilation fails due to an error at line 28
B. Compilation fails due to error at line 15 and 16

C. The contents of report1.txt are copied to consolidate.txt. The contents of report2.txt are appended to consolidate.txt, after a new line
D. The contents of report1.txt are copied to consolidate.txt. The contents of report2.txt are appended to consolidate.txt, without a break in the flow.

Answer: A
Explanation: The auto-closable resource fis may not be assigned.

Note: The try-with-resources statement is a try statement that declares one or more resources. A resource is an object that must be closed after the program is finished with it. The try-with-resources statement ensures that each resource is closed at the end of the statement. Any object that implements java.lang.AutoCloseable, which includes all objects which implement java.io.Closeable, can be used as a resource.

Reference: The Java Tutorials,The try-with-resources Statement

QUESTION NO: 29

How many Threads are created when passing tasks to an Executor Instance?

A. A new Thread is used for each task.
B. A number of Threads equal to the number of CPUs is used to execute tasks.
C. A single Thread is used to execute all tasks.
D. A developer-defined number of Threads Is used to execute tasks.
E. A number of Threads determined by system load is used to execute tasks.
F. The method used to obtain the Executor determines how many Threads are used to execute tasks.

Answer: F
Explanation: A simple way to create an executor that uses a fixed thread pool is to invoke the newFixedThreadPool factory method in java.util.concurrent.Executors This class also provides the following factory methods:
* The newCachedThreadPool method creates an executor with an expandable thread pool. This executor is suitable for applications that launch many short-lived tasks.
* The newSingleThreadExecutor method creates an executor that executes a single task at a time.
* Several factory methods are ScheduledExecutorService versions of the above executors.

If none of the executors provided by the above factory methods meet your needs, constructing instances of java.util.concurrent.ThreadPoolExecutor or

java.util.concurrent.ScheduledThreadPoolExecutor will give you additional options.

Note:The Executor interface provides a single method, execute, designed to be a drop-in replacementfor a common thread-creation idiom. If r is a Runnable object, and e is an Executor object you can replace
(new Thread(r)).start();
with
e.execute(r);
However, the definition of execute is less specific. The low-level idiom creates a new thread and launches it immediately. Depending on the Executor implementation, execute may do the same thing, but is more likely to use an existing worker thread to run r, or to place r in a queue to wait for a worker thread to become available.

Reference: The Java Tutorials,Thread Pools

Reference: The Java Tutorials,Executor Interfaces

QUESTION NO: 30

Given the code fragment:

SimpleDateFormat sdf;

Which code fragment displays the three-character month abbreviation?

A. sdf = new SimpleDateFormat ("mm", Locale.UK);
System.out.println ("Result:"+ sdf.format(new Date()));
B. sdf = new SimpleDateFormat (“MM”, Locale.UK);
System.out.println ("Result:"+ sdf.format(new Date()));
C. sdf = new SimpleDateFormat (“MMM”, Locale.UK);
System.out.println ("Result:"+ sdf.format(new Date()));
D. sdf = new SimpleDateFormat (“MMMM”, Locale.UK);
System.out.println ("Result:"+ sdf.format(new Date()));

Answer: C
Explanation: C: Output example: Apr

Note:SimpleDateFormat is a concrete class for formatting and parsing dates in a locale-sensitive manner. It allows for formatting (date -> text), parsing (text -> date), and normalization. SimpleDateFormat allows you to start by choosing any user-defined patterns for date-time

formatting. However, you are encouraged to create a date-time formatter with either getTimeInstance, getDateInstance, orgetDateTimeInstance in DateFormat. Each of these class methods can return a date/time formatter initialized with a default format pattern. You may modify the format pattern using the applyPattern methods as desired.

QUESTION NO: 31

Given the code fragment:

```
public static void processFile () throws IOException
{ Try (FileReader fr = new FileReader
("logfilesrc.txt"); FileWriter fw = new FileWriter
("logfilesdst.txt") ) {
int i = fr.read();
}
}
```

Which statement is true?

A. The code fragment contains compilation errors.
B. The java runtime automatically closes the FileWriter Instance first and the FileReader instance next.
C. The java runtime automatically closes the FileReader Instance first and the FileWriter instance next.
D. The developer needs to close the FileReader instance first and the FileWriter instance explicitly in a catch block.
E. The Java runtime may close the FileReader and FileWriter instance in an intermediate manner. Developers should not rely on the order in which they are closed.

Answer: B
Explanation: The try-with-resources statement is a try statement that declares one or more resources. Aresource is an object that must be closed after the program is finished with it. The try-with-resources statement ensures that each resource is closed at the end of the statement. Any object that implements java.lang.AutoCloseable, which includes all objects which implementjava.io.Closeable, can be used as a resource.

Reference: The Java Tutorials,The try-with-resources Statement

QUESTION NO: 32

Given this code fragment:

```
ResultSet rs = null;

try (Connection conn = DriverManager. getConnection (url) )

{ Statement stmt = conn.createStatement();

rs  stmt.executeQuery(query);

//-.. other methods }

} catch  (SQLException se)

{ System.out.println  ("Error");

}
```

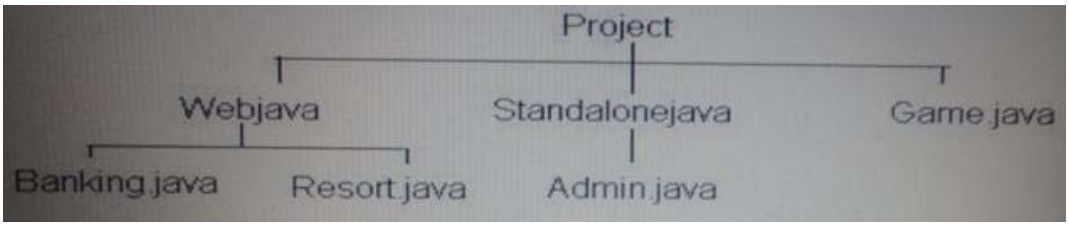
Which object is valid after the try block runs?

A. The Connection object only
B. The Statement object only
C. The Result set object only
D. The Statement and Result Set object only
E. The connection, statement, and ResultSet objects
F. Neither the Connection, Statement, nor ResultSet objects

Answer: C
Explanation: Generally, JavaScript has just 2 levels of scope: global and function. But, try/catch is an exception (no punn intended). When an exception is thrown and the exception object gets a variable assigned to it, that object variable is only available within the "catch" section and is destroyed as soon as the catch completes.

QUESTION NO: 33

View the Exhibit:



Given the following code fragment:

```
class Finder extends SimpleFileVisitor<Path> {
private final PathMatcher matcher;
private static int numMatches = 0;

Finder() {
matcher = FileSystems.getDefault().getPathMatcher("glob:*java");
}

void find(Path file) {
Path Name = file.getFileName();
if (name != null && matcher.matches(name)) {
numMatches++;
}
}
void report()
{
System.out.println("Matched: " + numMatches);
}
@Override
public FileVisitResult visitFile(Path file, BasicFileAttributes attrs) {
find(file);
return CONTINUE;
}
```

```
}

public class Visitor {

public static void main(String[] args) throws IOException

{ Finder finder = new Finder();

Files.walkFileTree(Paths.get("d:\\Project"), finder);

finder.report();

}

}
```

What is the result?

A. Compilation fails
B. 6
C. 4
D. 1
E. 3

Answer: B
Explanation: The program will compile and run.
Referring to the exhibit there will be six nodes that matches glob:*java.

QUESTION NO: 34

Given the following code fragment:

```
public static void main(String[] args)

{ Path tempFile = null;

try {

Path p = Paths.get("emp");

tempFile = Files.createTempFile(p, "report", ".tmp");
```

```
try (BufferedWriter writer = Files.newBufferedWriter(tempFile, Charset.forName("UTF8"))){
writer.write("Java SE 7");
}
System.out.println("Temporary file write done");
} catch(IOException e) {
System.err.println("Caught IOException: " + e.getMessage());
}
}
```

What is the result?

A. The report.tmp file is purged during reboot.
B. The report.tmp file is automatically purged when it is closed.
C. The report.tmp file exists until it is explicitly deleted.
D. The report.tmp file is automatically purged when the execution of the program completes.

Answer: C
Explanation: The createTempFile (String prefix,
String suffix,
FileAttribute<?>... attrs) method creates an empty file in the default temporary-file directory, using the given prefix and suffix to generate its name.

This method is only part of a temporary-file facility. Where used as a work files, the resulting file may be opened using the DELETE_ON_CLOSE option so that the file is deleted when the appropriate close method is invoked. Alternatively, a shutdown-hook, or the File.deleteOnExit() mechanism may be used to delete the file automatically.
In this scenario no delete mechanism is specified.

Reference: java.nio.file.createTempFile

QUESTION NO: 35

Which two Capabilities does Java.util.concurcent.BlockingQueue provide to handle operation that

cannot be handled immediately?

A. Automatically retry access to the queue with a given periodicity.
B. Wait for the queue to contain elements before retrieving an element.
C. Increase the queue's capacity based on the rate of blocked access attempts.
D. Wait for space to become available in the queue before inserting an element.

Answer: B,D
Explanation: A blocking queue is a Queue that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

Note: The BlockingQueue interface in the java.util.concurrent class represents a queue which is thread safe to put into, and take instances from.

The producing thread will keep producing new objects and insert them into the queue, until the queue reaches some upper bound on what it can contain. It's limit, in other words. If the blocking queue reaches its upper limit, the producing thread is blocked while trying to insert the new object. It remains blocked until a consuming thread takes an object out of the queue.

The consuming thread keeps taking objects out of the blocking queue, and processes them. If the consuming thread tries to take an object out of an empty queue, the consuming thread is blocked until a producing thread puts an object into the queue.

Reference: Java.util.concurcent.BlockingQueue

QUESTION NO: 36

Which is true regarding the java.nio.file.Path Interface?

A. The interface extends WatchService interface
B. Implementations of this interface are immutable.
C. Implement at ions of this interface arc not safe for use by multiple concurrent threads.
D. Paths associated with the default provider are not interoperable with the

Answer: A
Explanation: The java.nio.file.Path interface extends Watchable interface so that a directory located by a path can be registered with a WatchService and entries in the directory watched.

Note: An object that may be used to locate a file in a file system. It will typically represent a system

dependent file path.
A Path represents a path that is hierarchical and composed of a sequence of directory and file name elements separated by a special separator or delimiter. A root component, that identifies a file system hierarchy, may also be present. The name element that is farthest from the root of the directory hierarchy is the name of a file or directory. The other name elements are directory names. A Path can represent a root, a root and a sequence of names, or simply one or more name elements. A Path is considered to be an empty path if it consists solely of one name element that is empty. Accessing a file using an empty path is equivalent to accessing the default directory of the file system. Path defines the getFileName, getParent, getRoot, and subpath methods to access the path components or a subsequence of its name elements.

Reference: java.nio.file.Path Interface

QUESTION NO: 37

What are two benefits of a Factory design pattern?

A. Eliminates direct constructor calls in favor of invoking a method
B. Provides a mechanism to monitor objects for changes
C. Eliminates the need to overload constructors in a class implementation
D. Prevents the compile from complaining about abstract method signatures
E. Prevents tight coupling between your application and a class implementation

Answer: A,E
Explanation: Factory methods are static methods that return an instance of the native class.
Factory methods :
* have names, unlike constructors, which can clarify code.
* do not need to create a new object upon each invocation - objects can be cached and reused, if necessary.
* can return a subtype of their return type - in particular, can return an object whose implementation class is unknown to the caller. This is a very valuable and widely used feature in many frameworks which use interfaces as the return type of static factory methods.

Note: The factory pattern (also known as the factory method pattern) is a creational design pattern. A factory is a JavaSW class that is used to encapsulate object creation code. A factory class instantiates and returns a particular type of object based on data passed to the factory. The different types of objects that are returned from a factory typically are subclasses of a common parent class.
The data passed from the calling code to the factory can be passed either when the factory is created or when the method on the factory is called to create an object. This creational method is

often called something such as *getInstance* or *getClass* .

QUESTION NO: 38

The two methods of course rescue that aggregate the features located in multiple classes are _____.

A. Inheritance
B. Copy and Paste
C. Composition
D. Refactoring
E. Virtual Method Invocation

Answer: C,E
Explanation: C: Composition is a special case of aggregation. In a more specific manner, a restricted aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition.
E: In object-oriented programming, a virtual function or virtual method is a function or method whose behaviour can be overridden within an inheriting class by a function with the same signature. This concept is a very important part of the polymorphism portion of object-oriented programming (OOP).
The concept of the virtual function solves the following problem:
In OOP when a derived class inherits a base class, an object of the derived class may be referred to (or cast) as either being the base class type or the derived class type. If there are base class methods overridden by the derived class, the method call behaviour is ambiguous.
The distinction between virtual and non-virtual resolves this ambiguity. If the function in question is designated virtual in the base class then the derived class' function would be called (if it exists). If it is not virtual, the base class' function would be called.
Virtual functions overcome the problems with the type-field solution by allowing the programmer to declare functions in a base class that can be redefined in each derived class.

Note:Aggregation is a special case of association. A directional association between objects. When an object ‘has-a’ another object, then you have got an aggregation between them. Direction between them specified which object contains the other object. Aggregation is also called a “Has-a” relationship.

Trying our product !

★ **100%** Guaranteed Success

★ **100%** Money Back Guarantee

★ **365 Days** Free Update

★ **Instant Download** After Purchase

★ **24x7** Customer Support

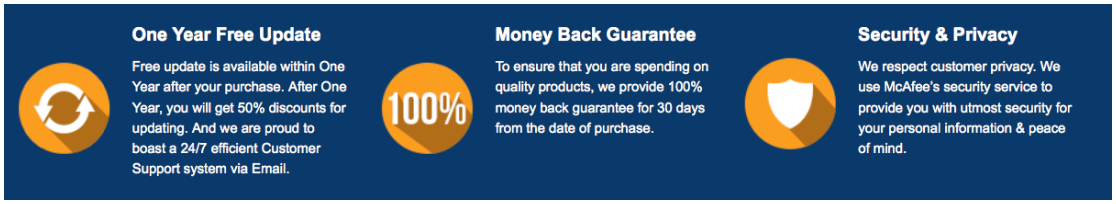
★ Average **99.9%** Success Rate

★ More than **69,000** Satisfied Customers Worldwide

★ Multi-Platform capabilities - **Windows, Mac, Android, iPhone, iPod, iPad, Kindle**

Need Help

Please provide as much detail as possible so we can best assist you.
To update a previously submitted ticket:



Guarantee & Policy | Privacy & Policy | Terms & Conditions