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");

}
```

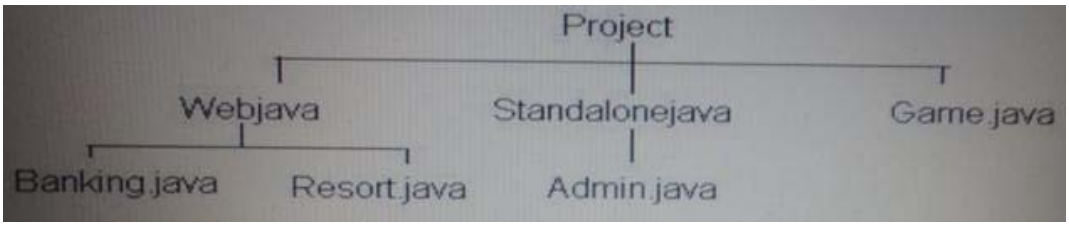
Which object is valid after the try block runs?

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.

QUESTION NO: 39

Given:

```
public class DAOManager {

public AccountDAO getAccountDAO()  {

return new AccountJDBCDAO();


}

}
```

Which design pattern best describes the class?

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

Answer: B
Explanation: Data Access Object

Abstracts and encapsulates all access to a data source
Manages the connection to the data source to obtain and store data
Makes the code independent of the data sources and data vendors (e.g. plain-text, xml, LDAP, MySQL, Oracle, DB2)

QUESTION NO: 40

Which two scenarios throw FileSystemNotFoundException when the paths.get(URI) method is invoked?

A. When preconditions on the uri parameter do not hold
B. When the provider identified by the URI'S scheme component is not installed
C. When a security manager is installed and it denies an unspecified permission to acess the file system.
D. When the file system identified by the uri does not exist and cannot be created automatically
E. When the path string cannot be converted to a Path

Answer: B,D

Explanation: The file system, identified by the URI, does not exist and cannot be created automatically, or the provider identified by the URI's scheme component is not installed.

Note: This method converts the given URI to a Path object.
It throws the following exceptions:
*IllegalArgumentException - if preconditions on the uri parameter do not hold. The format of the URI is provider specific.
*FileSystemNotFoundException - The file system, identified by the URI, does not exist and cannot be created automatically, or the provider identified by the URI's scheme component is not installed
*SecurityException - if a security manager is installed and it denies an unspecified permission to access the file system

Reference: java.nio.file.Paths

QUESTION NO: 41

Which code fragment is required to load a JDBC 3.0 driver?

A. DriverManager.loadDriver(“org.xyzdata.jdbc.network driver”);
B. Class.forname (“org.xyzdata.jdbc.NetWorkDriver”);
C. Connection con = Connection.getDriver
(“jdbc:xyzdata: //localhost:3306/EmployeeDB”);
D. Connection con = Drivermanager.getConnection
(“jdbc:xyzdata: //localhost:3306/EmployeeDB”);

Answer: B
Explanation: Note that your application must manually load any JDBC drivers prior to version 4.0.
The simplest way to load a driver class is to call the Class.forName() method.
If the fully-qualified name of a class is available, it is possible to get the corresponding Class using the static method Class.forName().

QUESTION NO: 42

What statement is true about thread starvation?

A. Thread "A" is said to be starved when it is frequently unable to gain access to a resource that it shares with another thread.

B. Thread "A" is said to be starved when it is blocked waiting for Thread "13," which in turn waiting for Thread "A."
C. Starvation can occur when threads become so busy responding to other threads that they move forward with their other work.
D. When some of the processors in a multi-processor environment go offline and the live thread(s) blocked waiting for CPU cycles, that blocking is referred to as "thread starvation."

Answer: A
Explanation: Starvation describes a situation where a thread is unable to gain regular access to shared resources and is unable to make progress. This happens when shared resources are made unavailable for long periods by "greedy" threads. For example, suppose an object provides a synchronized method that often takes a long time to return. If one thread invokes this methodfrequently, other threads that also need frequent synchronized access to the same object will often be blocked.

Reference: The Java Tutorials,Starvation and Livelock

QUESTION NO: 43

Given the code fragment:

```
static void addContent () throws Exception {

Path path = Paths.get("D:\\company\\report.txt");

UserPrincipal owner =
path.getFileSystem().getUserPrincipalLookupService().lookupPrincipalByName("Bob");

Files.setOwner(path, owner);

// insert code here – Line **

br.write("this is a text message ");

}

System.out.println("success");

}
```

Assume that the report.txt file exists.

Which try statement, when inserted at line **, enables appending the file content without writing the metadata to the underlying disk?

A. try (BufferWriter br = Files.newBufferedWriter (path, Charset.forName ("UTF-8"), new openOption []
{StandardOpenOption.CREATE, StandardOpenOption.Append, StandardOpenOption.DSYNC}};}
{
B. try (BufferWriter br = Files.newBufferedWriter (path, Charset.forName ("UTF-8"), new openOption [] {StandardOpenOption.APPEND, StandardOpenOption.SYNC));){
C. try (BufferWriter br = Files.newBufferedWriter (path, Charset.forName ("UTF - 8"), new openOption [] {StandardOpenOption.APPEND, StandardOpenOption.DSYNC}
D. try (BufferWriter br = Files.newBufferedWriter (path, Charset.forName ("UTF–8"), new openOption []
{StandardOpenOption.CREATENEW, StandardOpenOption.APPEND, StandardOpenOption.SYNC}}
}
E. try (BufferWriter br = Files.newBufferedWriter (path, Charset.forName (“UTF - 8”), new openOption [] {StandardOpenOption.APPEND, StandardOpenOption.ASYNC});) {

Answer: C
Explanation: StandardOpenOption should be both APPEND (if the file is opened for WRITE access then bytes will be written to the end of the file rather than the beginning)and DSYNC (Requires that every update to the file's content be written synchronously to the underlying storage device.).

Note 1:The newBufferedWriter method Opens or creates a file for writing, returning a BufferedWriter that may be used to write text to the file in an efficient manner. The options parameter specifies how the the file is created or opened. If no options are present then this method works as if the CREATE, TRUNCATE_EXISTING, and WRITE options are present. In other words, it opens the file for writing, creating the file if it doesn't exist, or initially truncating an existing regular-file to a size of 0 if it exists.

Note2: public static final StandardOpenOption APPEND
If the file is opened for WRITE access then bytes will be written to the end of the file rather than the beginning.
If the file is opened for write access by other programs, then it is file system specific if writing to the end of the file is atomic.

Reference: java.nio.file.Files

java.nio.file Enum StandardOpenOption

QUESTION NO: 44

Which two statements are true?

A. Implementing a DAO often includes the use of factory.
B. To be implemented properly, factories rely on the private keyword.
C. Factories are an example of the OO principle "program to an interface."
D. Using factory prevents your replication from being tightly coupled with a specific Singleton
E. One step in implementing a factory is to add references from all the classes that the factory will merge.

Answer: A,C
Explanation: A: 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.

C: The essence of the Factory method Pattern is to "Define an interface for creating an object, but let the classes which implement the interface decide which class to instantiate. The Factory method lets a class defer instantiation to subclasses."

Note:The factory method pattern is an object-oriented design pattern to implement the concept of factories. Like other creational patterns, it deals with the problem of creating objects (products) without specifying the exact class of object that will be created. The creation of an object often requires complex processes not appropriate to include within a composing object. The object's creation may lead to a significant duplication of code, may require information not accessible to the composing object, may not provide a sufficient level of abstraction, or may otherwise not be part of the composing object's concerns. The factory method design pattern handles these problems by defining a separate method for creating the objects, whichsubclasses can then override to specify the derived type of product that will be created.
Some of the processes required in the creation of an object include determining which object to create, managing the lifetime of the object, and managing specialized build-up and tear-down concerns of the object.

QUESTION NO: 45

Which five items are provided by the Java concurrency utilities?

A. High-performance, flexible thread pools
B. Dynamic adjustment of thread priorities
C. Collection classes designed for concurrent access
D. Atomic variables
E. synchronized wrappers for collection classes in the java.util package,
F. Asynchronous execution of tasks
G. Counting semaphores
H. Concurrent collection sorting implementations

Answer: A,C,D,E,G
Explanation: The Java 2 platform includes a new package of concurrency utilities. These are classes that are designed to be used as building blocks in building concurrent classes or applications. Just as thecollections framework simplified the organization and manipulation of in-memory data by providing implementations of commonly used data structures, the concurrency utilities simplify the development of concurrent classes by providing implementations of building blocks commonly used in concurrent designs. The concurrency utilities include a high-performance, flexible thread pool; a framework for asynchronous execution of tasks; a host of collection classes optimized for concurrent access; synchronization utilities such as counting semaphores (G); atomic variables; locks; and condition variables.

The concurrency utilities includes:

* Task scheduling framework. The Executor interface standardizes invocation, scheduling, execution, and control of asynchronous tasks according to a set of execution policies. Implementations are provided that enable tasks to be executed within the submitting thread, in a single background thread (as with events in Swing), in a newly created thread, or in a thread pool, and developers can create customized implementations of Executor that support arbitrary execution policies. The built-in implementations offer configurable policies such as queue length limits and saturation policy that can improve the stability of applications by preventing runaway resource use.
* Fork/join framework. Based on the ForkJoinPool class, this framework is an implementation of Executor. It is designed to efficiently run a large number of tasks using a pool of worker threads (A) . A work-stealing technique is used to keep all the worker threads busy, to take full advantage of multiple processors.
* (C) Concurrent collections. Several new collections classes were added, including the new Queue, BlockingQueue and BlockingDeque interfaces, and high-performance, concurrent implementations of Map, List, and Queue. See the Collections Framework Guide for more information.
* (D) Atomic variables. Utility classes are provided that atomically manipulate single variables (primitive types or references), providing high-performance atomic arithmetic and compare-and-set methods. The atomic variable implementations in the java.util.concurrent.atomic package offer higher performance than would be available by using synchronization (on most platforms), making them useful for implementing high-performance concurrent algorithms and conveniently

implementing counters and sequence number generators.
* (E) Synchronizers. General purpose synchronization classes, including semaphores, barriers, latches, phasers, and exchangers, facilitate coordination between threads.
* Locks. While locking is built into the Java language through the synchronized keyword, there are a number of limitations to built-in monitor locks. The java.util.concurrent.locks package provides a high-performance lock implementation with the same memory semantics as synchronization, and it also supports specifying a timeout when attempting to acquire a lock, multiple condition variables per lock, nonnested ("hand-over-hand") holding of multiple locks, and support for interrupting threads that are waiting to acquire a lock.
* Nanosecond-granularity timing. The System.nanoTime method enables access to a nanosecond-granularity time source for making relative time measurements and methods that accept timeouts (such as the BlockingQueue.offer, BlockingQueue.poll, Lock.tryLock, Condition.await, and Thread.sleep) can take timeout values in nanoseconds. The actual precision of the System.nanoTime method is platform-dependent.

Reference: Java SE Documentation, Concurrency Utilities

QUESTION NO: 46

Given:

```
import java.io.*;

public class SampleClass {

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

try {

String dirName = args[0];

File dir = new File(dirName);

File.createTempFile("temp", "log", dir);

} catch (NullPointerException | IOException e) {

e = new IOException("Error while creating temp file");

throw e;

}

}
```

}

What is the result?

A. Compilation fails.
B. An IOException with the default message is thrown at runtime.
C. An IOException with the message Error while creating temp file is thrown at runtime.
D. A temporary file is created in the specified directory.

Answer: A
Explanation: The multi-catch parameter e may not be assigned. The compilation will fail at line:
e = new IOException("Error while creating temp file");

QUESTION NO: 47

Given a language code of fr and a country code of FR, while file name represents a resource bundle file name that is not the default?

A. MessageBundle_fr_FR.properties
B. MessageBundle_fr_FR.Profile
C. MessageBundle_fr_FR.Xml **D.** MessageBundle_fr_FR.Java **E.** MessageBundle_fr_FR.locale

Answer: A
Explanation: A properties file is a simple text file. You can create and maintain a properties file with just about any text editor.
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.
To support an additional Locale, your localizers will create a new properties file that contains the translated values. No changes to your source code are required, because your program references the keys, not the values.

For example, to add support for the German language, your localizers would translate the values in LabelsBundle.properties and place them in a file named LabelsBundle_de.properties. Notice that the name of this file, like that of the default file, begins with the base name LabelsBundle and ends with the .properties suffix.

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

QUESTION NO: 48

Given:

Person

public Person(int id)

public int getid()

public String getContactDetails()

public void setContactDetails(String contactDetails)

public String getName()

public void setName(String name)

public Person getPerson(int id) throws Exception

public void createPerson(int id) throws Exception

public Person deletePerson(int id) throws Exception

public void updatePerson(Person p) throws Exception

Which group of methods is moved to a new class when implementing the DAO pattern?

A. public int getId ()
public String getContractDetails ()
public void setContactDetails (String ContactDetails)
public void getName ()
public Person setName (String name)
B. public int getId ()
public String getContractDetails ()
public void getName ()
public person getPerson (int id) throws Exception
C. public void setContactDetails(String contactDetails)
public void setName (String name)
D. public person getPerson(int id) throws Exception
public void createPerson (person p) throws exception
public void deleteperson(int id) throws Exception
public void updatePerson (Person p) throws Exception

Answer: D

Note:Data Access Object

Abstracts and encapsulates all access to a data source
Manages the connection to the data source to obtain and store data
Makes the code independent of the data sources and data vendors (e.g. plain-text, xml, LDAP, MySQL, Oracle, DB2)

QUESTION NO: 49

Given the code fragment:

```
SimpleDateFormat sdf = new SimpleDateFormat("zzzz",  Locale.US);

System.out.println ("Result:  " + sdf.format(today) ) ;
```

What type of result is printed?

A. Time zone abbreviation
B. Full-text time zone name
C. Era
D. Julian date
E. Time of the Epoch (in milliseconds)

Answer: A
Explanation: Assuming that the variable today contains a date, the time zone abbreviation, such as Pacific Standard Time or Central European Summer Time, will be printed.

QUESTION NO: 50

The advantage of a CallableStatement over a PreparedStatement is that it:

A. Is easier to construct
B. Supports transactions
C. Runs on the database

D. Uses Java instead of native SQL

Answer: C
Explanation: A Statement is an interface that represents a SQL statement. You execute Statement objects, and they generate ResultSet objects, which is a table of data representing a database result set.
There are three different kinds of statements:
* Statement: Used to implement simple SQL statements with no parameters.
* PreparedStatement: (Extends Statement.) Used for precompiling SQL statements that might contain input parameters.
* CallableStatement: (Extends PreparedStatement.) Used to execute stored procedures that may contain both input and output parameters.

A stored procedure is a group of SQL statements that form a logical unit and perform a particular task, and they are used to encapsulate a set of operations or queries to execute on a database server.

Reference: The Java Tutorials:

Executing Queries

Using Stored Procedures

QUESTION NO: 51

Given:

class Fibonacci extends RecursiveTask<Integer> {

final int n;

Fibonacci (int n) { this.n = n }

Integer compute () {

if (n <= 1)

return n;

Fibonacci f1 = new Fibonacci (n – 1);

f1.fork;

Fibonacci f2 = new Fibonacci (n – 2);

```
return f2.compute() + f1.join;  // Line **
}
}
```

Assume that line ** is replaced with:

```
return f1.join() + f2.compute();  // Line **
```

What is the likely result?

A. The program produces the correct result, with similar performance to the original.
B. The program produces the correct result, with performance degraded to the equivalent of being single-threaded.
C. The program produces an incorrect result.
D. The program goes into an infinite loop.
E. An exception is thrown at runtime.
F. The program produces the correct result, with better performance than the original.

Answer: B
Explanation: Changing the code is not useful. In the original code (return f2.compute() + f1.join;) f1 and f2 are run in parallel. The result is joined.
With the changed code (return f1.join() + f2.compute();) f1 is first executed and finished, then is f2 executed.

Note 1:The join method allows one thread to wait for the completion of another.
If t is a Thread object whose thread is currently executing,
t.join();
causes the current thread to pause execution until t's thread terminates.

Note 2:New in the Java SE 7 release, the fork/join framework is an implementation of the ExecutorService interface that helps you take advantage of multiple processors. It is designed for work that can be broken into smaller pieces recursively. The goal is to use all the available processing power to enhance the performance of your application.

As with any ExecutorService, the fork/join framework distributes tasks to worker threads in a thread pool. The fork/join framework is distinct because it uses a work-stealing algorithm. Worker threads that run out of things to do can steal tasks from other threads that are still busy.

Reference: The Java Tutorials,Joins,Fork/Join

QUESTION NO: 52

Given:

```
public static void main(String[] args) throws Exception {
try {
processFile();
} catch(Exception e) {
Throwable [] t = e.getSuppressed();
}
}

public static void processFile() throws IOException
{ try (FileReader fr = new FileReader"logfilesrc.txt");
FileWriter fw = new FileWriter("logfiledest.txt")) {{
java.util.zip.ZipFile zf = new java.util.zip.ZipFile("alllogs.zip");
System.out.println("Created files for logs");
}
}
```

The getSuppressed method returns an array of _____.

A. Allexceptions that were thrown in the processFile method but were suppressed.
B. Exceptions suppressed because that are not declared in the throws clause.
C. Only runtime exceptions that were thrown in the processFile method but were suppressed.
D. Only runtime exceptions that were thrown in the processFile method but were not declared in throws clause.

Answer: A
Explanation: The GetSuppressed() methodreturns an array containing all of the exceptions that were suppressed, typically by the try-with-resources statement, in order to deliver this exception. If an exception is thrown from the try block and one or more exceptions are thrown from the try-with-resources statement, then those exceptions thrown from the try-with-resources statement are suppressed.

Reference: The Java Tutorials,Suppressed Exceptions

QUESTION NO: 53

Given the code fragment:

```
11. public static getFileSize () throws IOException {
12. path file = paths.get ("ex.txt");
13. //insert code here
14. System.out.println ("size: " + attr.size());
15. }
```

```
public static getFileSize () throws IOException
{ Path file = Paths.get ("ex.txt");
//insert code here Line **
System.out.println ("size: " + attr.size());
}
```

Which two fragments, when inserted independently at line **, enable printing of the file size?

A. BasicFileAttributes attr = Files.readAttributes (file, BasicFileAttributes.class);
B. PosixFileAttributes attr = Files.readAttributes (file, posixFileAttributes.class);
C. DosFileAttributes attr = Files.readAttributes (file, dosAttributes.class);
D. FileStore attr = Files.getFileStore (file);
E. AclFileAttributeview attr = Files.getFileAttributeView(File, AclFileAttributeview.class);

Answer: A,B
Explanation: A: The BasicFileAttributes has a size method.
B: The PosixFileAttributes has a size method.

QUESTION NO: 54

Which two statements are true about RowSet subinterfaces?

A. A jdbcRowSet object provides a JavaBean view of a result set.
B. A cachedRowSet provides a connected view of the database.
C. A FilteredRowSet object filter can be modified at any time.
D. A webRowSet returns JSON-formatted data.

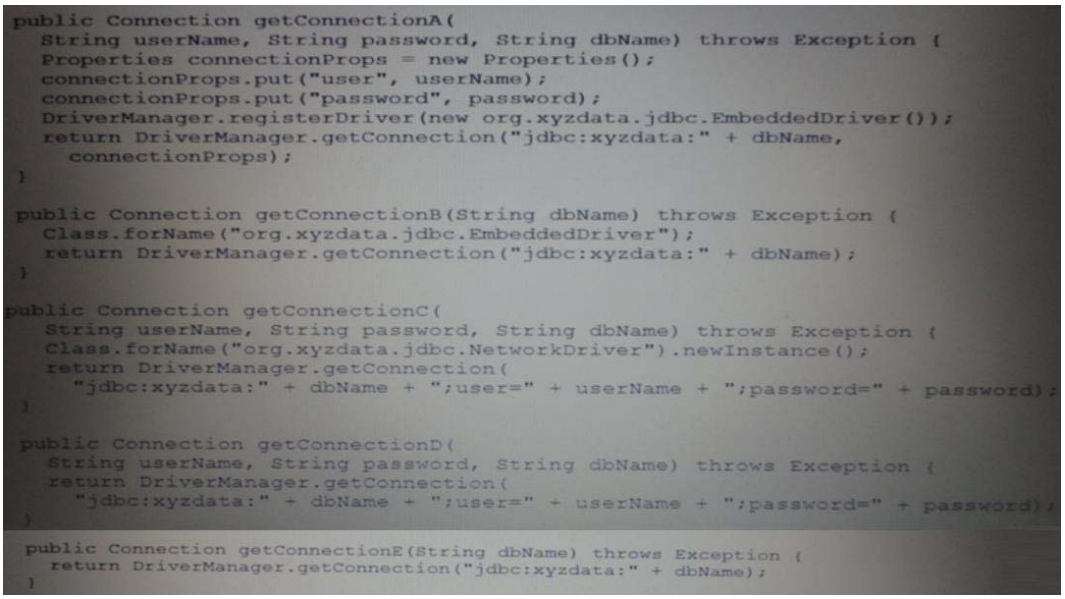
Answer: A,C
Explanation: A:JdbcRowSet Mmakes results available as a JavaBean component
C: FilteredRowSetimplements lightweight querying, usingjavax.sql.rowset.Predicate
The predicate set on a FilteredRowSet object applies a criterion on all rows in a RowSet object to manage a subset of rows in aRowSet object. This criterion governs the subset of rows that are visible and also defines which rows can be modified, deleted or inserted.
Therefore, the predicate set on a FilteredRowSet object must be considered as bi-directional and the set criterion as the gating mechanism for all views and updates to the FilteredRowSet object. Any attempt to update the FilteredRowSet that violates the criterion will result in a SQLException object being thrown.
The FilteredRowSet range criterion can be modified by applying a new Predicate object to the FilteredRowSet instance at any time. This is possible if no additional references to the FilteredRowSet object are detected. A new filter has has an immediate effect on criterion enforcement within the FilteredRowSet object, and all subsequent views and updates will be subject to similar enforcement.

Note: The RowSet interface, a subinterface of ResultSet , was introduced with JDBC 2. Because a RowSet is a ResultSet , you can use any of the ResultSet processing methods previously discussed. But RowSet s tend to be more self-contained; you typically do not need to specify a driver, and performing queries is done in a new way. You call setCommand() to specify the query and execute() to perform the query (this takes the place of creating a Statement and calling its executeQuery() method).
Incorrect answer:
B: CachedRowSetcaches results in memory; disconnected Rowset.
D:WebRowSet converts between XML data and RowSet. The data is not JSON formatted.

Note: JSON or JavaScript Object Notation, is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for many languages.

QUESTION NO: 55

Consider the following five methods:



```
public Connection getConnectionA(
  String userName, String password, String dbName) throws Exception {
  Properties connectionProps = new Properties();
  connectionProps.put("user", userName);
  connectionProps.put("password", password);
  DriverManager.registerDriver(new org.xyzdata.jdbc.EmbeddedDriver());
  return DriverManager.getConnection("jdbc:xyzdata:" + dbName,
    connectionProps);
}

public Connection getConnectionB(String dbName) throws Exception {
  Class.forName("org.xyzdata.jdbc.EmbeddedDriver");
  return DriverManager.getConnection("jdbc:xyzdata:" + dbName);
}

public Connection getConnectionC(
  String userName, String password, String dbName) throws Exception {
  Class.forName("org.xyzdata.jdbc.NetworkDriver").newInstance();
  return DriverManager.getConnection(
    "jdbc:xyzdata:" + dbName + ";user=" + userName + ";password=" + password);
}

public Connection getConnectionD(
  String userName, String password, String dbName) throws Exception {
  return DriverManager.getConnection(
    "jdbc:xyzdata:" + dbName + ";user=" + userName + ";password=" + password);
}

public Connection getConnectionE(String dbName) throws Exception {
  return DriverManager.getConnection("jdbc:xyzdata:" + dbName);
}
```

Which method should you use to connect to a java Db database with JDBC 4.0, but not with previous versions of JDBC?

A. getConnectionA
B. getConnectionB
C. getConnectionC
D. getConnectionD
E. getConnectionE

Answer: D

Explanation: Note on D not E: Prior to JDBC 4.0, we relied on the JDBC URL to define a data source connection. Now with JDBC 4.0, we can get a connection to any data source by simply supplying a set of parameters (such as host name and port number) to a standard connection factory mechanism. New methods were added to Connection and Statement interfaces to permit improved connection state tracking and greater flexibility when managing Statement objects in pool environments.

Note on that an embedded driver is no longer needed(not A, B, C):
Thanks to the Java SE Service Provider mechanism included in Mustang, Java developers no longer need to explicitly load JDBC drivers using code like Class.forName() to register a JDBC driver. The DriverManager class takes care of this by automatically locating a suitable driver when the DriverManager.getConnection() method is called. This feature is backward-compatible, so no changes are needed to the existing JDBC code.

In JDBC 4.0, we no longer need to explicitly load JDBC drivers using Class.forName(). When the method getConnection is called, the DriverManager will attempt to locate a suitable driver from among the JDBC drivers that were loaded at initialization and those loaded explicitly using the same class loader as the current application.

Assume that we need to connect to an Apache Derby database, since we will be using this in the sample application explained later in the article:
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
Connection conn =
DriverManager.getConnection(jdbcUrl, jdbcUser, jdbcPassword);
But in JDBC 4.0, we don't need the Class.forName() line. We can simply call getConnection() to get the database connection.

QUESTION NO: 56

Given the code fragment:

public static void main(String[] args)

{ Path dir = Paths.get("d:\\company");

// insert code here. Line **

for (Path entry: stream)

{ System.out.println(entry.getFileName()

);

}

} catch (IOException e) {

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

}

}

Which two try statements, when inserted at line 11, enable you to print files with the extensions.java, .htm, end and .jar.

A. try (DirectoryStream<path> stream =Files.newDirectoryStream(dir, "*.{java, htm, jar}")) {
B. try (DirectoryStream<path> stream =Files.newDirectoryStream(dir, "*.[java, htm, jar"] }} {
C. try (DirectoryStream<path> stream =Files.newDirectoryStream(dir, "*.{java*, htm*, jar*}"}} {
D. try (DirectoryStream<path> stream =Files.newDirectoryStream(dir, "**.{java, htm, jar}")) {

Answer: A,D
Explanation:

Reference: The Java Tutorials

Finding Files

What *Is* a Glob?

QUESTION NO: 57

Given the code fragment:

Locale loc1 = Locale.getDefault ();

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

Which two statements are a valid way to re-assign a resource bundle to a different Locale?

A. loc1 = ResourceBundle.getBundle ("MessageBundle", Locale.CHINA);
B. loc1 = ResourceBundle.getBundle ("MessageBundle", new Locale ("es", "ES"));
C. messages = ResourceBundle.getBundle ("messageBundle", new Locale ("es", "ES"));
D. messages = ResourceBundle.getBundle ("MessageBundle", Locale.CHINA);

Answer: C,D

QUESTION NO: 58

Given:

public class TemperatureSensor {

public TemperatureSensor () {

}

public double getCurrTemp () {

// . . . method to retrieve temperature from a sensor

Return temp;

}

}

Which three changes should you make to apply the singleton design pattern to this class?

A. Make the class abstract.
B. Add a method to return a singleton class type.
C. Change the access of the constructor to private.
D. Add a public constructor that takes a single argument.
E. Change the class to implement the singleton interface.
F. Add a private static final variable that is initialized to new TemperatureSensor{};
G. Add a public static method that returns the class variable of type

Answer: C,F,G

Explanation: C: We provide a default Private constructor
F,G:We write a public static getter or access method(G)to get the instance of the Singleton Object at runtime. First time the object is created inside this method as it is null. Subsequent calls to this method returns the same object created as the object is globally declared (private)(F)and the hence the same referenced object is returned.
Note: 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 (C), 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: 59

Given the error message when running you application:

Exception in thread "main" java.util.MissingResourceException: can't find bundle for base name messageBundle, Locale

And given that the Message Bundle.properties file has been created, exists on your disk, and is properly formatted.

What is the cause of the error message?

A. The file is not in the environment PATH.
B. The file is not in the CLASSPATH.
C. The file is not in the JJAVAPATH.
D. You cannot use a file to store a ResourceBundle.

Answer: C
Explanation: Your language file should be in the classpath.

QUESTION NO: 60

Which two statements are true regarding the try with resources statement?

A. The resources declared in a try with resources statement are not closed automatically if an exception occurs inside the try block.
B. In a try with resources statement, any catch or finally block is run after the resources have been closed.
C. The close methods of resources are called in the reverse order of their creation.
D. All the resources must implement the java.io.closeable interface.

Answer: B,D
Explanation: B: Prior to Java SE 7, you can use a finally block to ensure that a resource is closed regardless of whether the try statement completes normally or abruptly.
A try-with-resources statement can have catch and finally blocks just like an ordinary trystateme
In a try-with-resources statement, any catch or finally block is run after the resources declared have been closed.

D: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: 61

Given the code fragment:

```
public class TestString {

public static void main(String[] args)

{ String str=null;
```

```
switch(str) { case "":

System.out.println("blank"); break;

case "null":

System.out.println("NULL"); break;

default: System.out.println("invalid");

break;

}

}

}
```

What is the result?

A. Compilation fails
B. Blank
C. NULL
D. An exception is thrown at runtime
E. Invalid

Answer: D
Explanation: A java.lang.NullPointerException will be thrown at runtime at line:
switch(str) {

Ensure that the expression in any switch statement is not null to prevent a NullPointerException from being thrown.

Reference: The Java Tutorials, The switch Statement

QUESTION NO: 62

Given the code fragment:

```
try {

String query = "SELECT * FROM Employee WHERE ID=110";
```

```
Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query);  // Line 13

System.out.println("Employee ID: " + rs.getInt("ID"));  // Line 14

} catch (Exception se)

{ System.out.println("Error"

);

}
```

Assume that the SQL query matches one record. What is the result of compiling and executing this code?

A. The code prints error.
B. The code prints the employee ID.
C. Compilation fails due to an error at line 13.
D. Compilation fails due to an error at line 14.

Answer: B
Explanation: Assuming that the connection conn has been set up fine, the code will compile and run fine.

Note#1: The GetInt method retrieves the value of the designated column in the current row of this ResultSet object as an int in the Java programming language.

Note 2: A table of data representing a database result set, which is usually generated by executing a statement that queries the database.

A ResultSet object maintains a cursor pointing to its current row of data. Initially the cursor is positioned before the first row. The next method moves the cursor to the next row, and because it returns false when there are no more rows in the ResultSet object, it can be used in a while loop to iterate through the result set.

A default ResultSet object is not updatable and has a cursor that moves forward only. Thus, you can iterate through it only once and only from the first row to the last row. It is possible to produce ResultSet objects that are scrollable and/or updatable.

Reference: The Java Tutorials,Interface ResultSet

QUESTION NO: 63

Given the code fragment:

```
Path path1 = Paths.get("D:\\sales\\.\\quarterly\\..\\report");
path1 = path1.normalize();

Path path2 = path1.relativize(Paths.get("d:\\empdetails.dat"));
path2 = path2.resolve(path1);

System.out.println(path1);
System.out.println(path2);
}
```

What is the result?

A. D: \sales\report
B. \sales\report
C. D: \sales\quarterly\ . . . \report
D. \sales\report
E. D: \sales\quarterly\ . . .\report
F. \sales\report\empdetails.dat
G. D: \sales\report
H. \sales\report\empdetails.dat

Answer: A
Explanation: Path1 is the normalized result of D:\\sales\\.\\quarterly\\..\\report
namely D: \sales\report.
The normalize method removes any redundant elements, which includes any "." or "directory/.." occurrences.

Consider path2.
With the relativize line path2 is set to**../../empdetails.dat**

In this scenario the following applies to the resolve statement: Passing an absolute path to the resolve method returns the passed-in path.So Path2 will be set to Path1 in the statementpath2 = path2.resolve(path1);

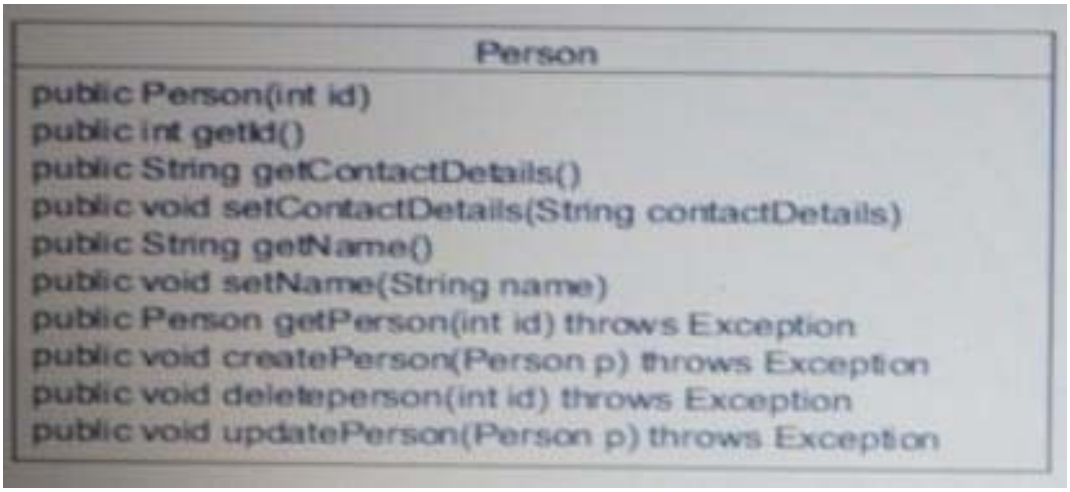
Note:
A common requirement when you are writing file I/O code is the capability to construct a path from one location in the file system to another location. You can meet this using the relativizemethod. This method constructs a path originating from the original path and ending at the location specified by the passed-in path. The new path is relative to the original path.

You can combine paths by using the resolve method. You pass in a partial path , which is a path that does not include a root element, and that partial path is appended to the original path.

Reference: The Java Tutorials,Path Operations

QUESTION NO: 64

Given:



Which design pattern moves the getPerson, createPerson, deletePerson, and updatePerson methods to a new class?

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

Answer: B
Explanation: We move the most abstract highest level methods into a separate class.

Abstracts and encapsulates all access to a data source
Manages the connection to the data source to obtain and store data
Makes the code independent of the data sources and data vendors (e.g. plain-text, xml, LDAP, MySQL, Oracle, DB2)

QUESTION NO: 65

Consider the following database table:

Inventory Table

* Item_ID, Integer: PK

* Item_name, Varchar (20)

* Price, Numeric (10, 2)

* Quan, Integer

Consider the following method that updates the prices in the Inventory table:

```
public static void updatePrices{
//  #1: missing line

Connection con)  throws SQLException {

//  #2: missing line

PreparedStatement updatePrices = con.prepareStatement (updatePricesString);

//  #3: missing line

{

//  #4: missing line

updatePrices.executeUpdate();
```

}

}

This method us missing four lines, which group of lines complete this method?

A. 1. HashMap<Integer, String> newPrices,
2. StringupdatePriceString ="UPDATE inventory SET price =? WHERE item_name'?'";
3. For (map.Entry<Integer, String> x : newPrices.entrySet())
4. UpdatePrices.setFloat(1, x.getvalue().floatValue()); updatePrice.setString (2, x.getKey());
B. 1. HashMap<Integer, Float> newPrices,
2. StringupdatePriceString ="UPDATE inventory SET price =? WHERE item_id '?'";
3. For (map.Entry<Integer, String> x : newPrices.entrySet())
4. UpdatePrices.setFloat(1, x.getvalue().floatValue()); updatePrice.setString (2, x.getKey().intValue());
C. 1. HashMap<Integer, Float> newPrices,
2. StringupdatePriceString = “UPDATE inventory SET price =? Where item_id ‘?’ ”;
3. For (map.Entry<Integer, String> x : newPrices.entrySet())
4. UpdatePrices.setInt(1, x.getvalue().floatValue()); updatePrice.setString (2, x.getvalue());
D. 1. HashMap<Integer, Float> newPrices,
2. StringupdatePriceString = “UPDATE inventory SET price =? Where item_id ‘?’ ”;
3. For (map.Entry<Integer, String> x : newPrices.entrySet())
4. UpdatePrices.setInt(1, x.getvalue().floatValue()); updatePrice.setString (2, x.getvalue().floatValue()
E. 1. HashMap<Integer, String> newPrices,
2. StringupdatePriceString = “UPDATE inventory SET price =? Where item_id ‘?’ ”;
3. For (map.Entry<Integer, String> x : newPrices.entrySet())
4. UpdatePrices,setString(1, x.getKey()); updatePrices.setFloat(2, x.getValue().floatValue());
F. 1. HashMap<Integer> newPrices,
2. StringupdatePriceString = “UPDATE inventory SET price =? Where item_id ‘?’ ”;
3. For (Integer x: newPrice)
4. updatePrice.setInt(1, x);

Answer: D
Explanation: The first line should be HashMap<Integer, Float> newPrices,
as in SQL numeric represent a float number, not an integeror string.
We also make sure to use floatValue() both in appropriate places in line 4.

Note: Map is an object that maps keys to values. A map cannot contain duplicate keys: Each key can map to at most one value. It models the mathematical function abstraction.

QUESTION NO: 66

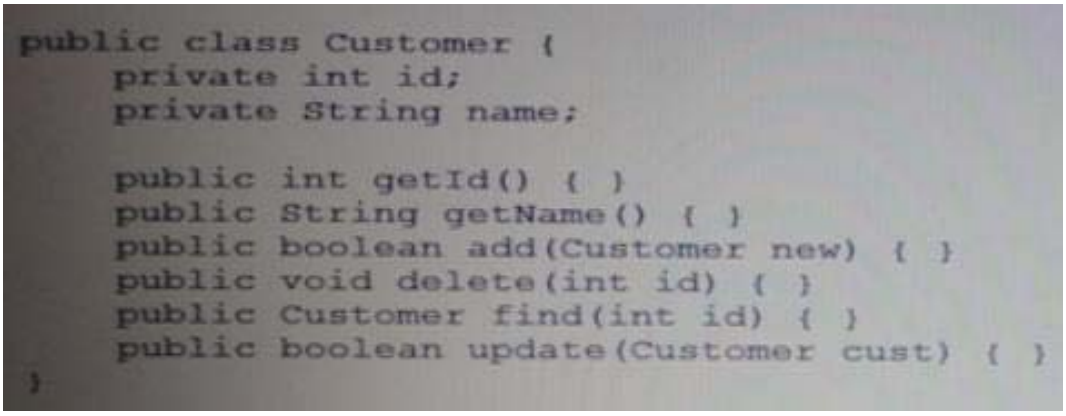
Which is a key aspect of composition?

A. Using inheritance
B. Method delegation
C. Creating abstract classes
D. Implementing the composite interface

Answer: B
Explanation: In an object-oriented design of a Java program, the way in which you model objects that contain other objects is with composition, the act of composing a class out of references to other objects. With composition, references to the constituent objects become fields of the containing object.
To use composition in Java, you use instance variables of one object to hold references to other objects.
The relationship modeled by composition is often referred to as the "has-a" relationship.
Delegation involves re-exporting methods; in a composition relationship, the inner objects methods may be used only privately and not re-exposed.

QUESTION NO: 67

Given:



```
public class Customer {
    private int id;
    private String name;

    public int getId() { }
    public String getName() { }
    public boolean add(Customer new) { }
    public void delete(int id) { }
    public Customer find(int id) { }
    public boolean update(Customer cust) { }
}
```

What two changes should you make to apply the DAO pattern to this class?

A. Make the customer class abstract.
B. Make the customer class an interface.
C. Move the add, delete, find, and update methods into their own implementation class.
D. Create an interface that defines the signatures of the add, delete, find and update command.
E. Make the add, delete, find, and update methods private for encapsulation.
F. Make the getName and getId methods private fir encapsulation.

Answer: C,D
Explanation: In computer software, a data access object (DAO) is an object that provides an abstract interface to some type of database or persistence mechanism, providing some specific operations without exposing details of the database. It provides a mapping from application calls to the persistence layer. This isolation separates the concerns of what data accesses the application needs, in terms of domain-specific objects and data types (the public interface of the DAO), and how these needs can be satisfied with a specific DBMS, database schema, etc. (the implementation of the DAO).

In the specific context of the Java programming language, Data Access Objects as a design concept can be implemented in a number of ways. This can range from a fairly simple interface that separates the data access parts from the application logic, to frameworks and commercial products.

QUESTION NO: 68

Which two descriptions are benefits of using PreparedStatement objects over static SQL in JDBC?

A. Conversion to native SQL
B. Supports BLOB types on every type of database
C. Prevention of SQL injection attacks
D. Improved performance from frequently run SQL queries
E. Built in support for multi database transaction semantics

Answer: A,D
Explanation: Sometimes it is more convenient to use a PreparedStatement object for sending SQL statements to the database. This special type of statement is derived from the more general class, Statement, that you already know.

If you want to execute a Statement object many times, it usually reduces execution time to use a PreparedStatement object instead.

The main feature of a PreparedStatement object is that, unlike a Statement object, it is given a SQL statement when it is created. The advantage to this is that in most cases, this SQL statement is sent to the DBMS right away, where it is compiled. As a result, the PreparedStatement object contains not just a SQL statement, but a SQL statement that has been precompiled. This means that when the PreparedStatement is executed, the DBMS can just run the PreparedStatement SQL statement without having to compile it first.

Although PreparedStatement objects can be used for SQL statements with no parameters, you

probably use them most often for SQL statements that take parameters. The advantage of using SQL statements that take parameters is that you can use the same statement and supply it with different values each time you execute it.

Reference: The Java Tutorials, Using Prepared Statements

QUESTION NO: 69

Given the code fragment:

```
/* method declaration */ {
try  {
String className = "java.lang.String";
String fieldname = "somefield";
Class c = Class.forName(className);
Field f = c.getField(fieldname);
} catch(Exception e)
{ e.printStackTrace();
throw e;
}
}
```

Which two method declarations are valid options to replace /* method declaration */?

A. public void getMetadata ()
B. public void getMetadat ()
C. public void getMetadata () throws Exception
D. public void getMetadata () throws NoSuchFieldException
E. public void getMetadata () throws classNotFoundException
F. public void getMetadata () throws ClassNotFoundException, NoSuchFieldException.

Answer: C,E
Explanation: We must specify that the getMetaData method can throw both ClassNotFoundException (line Class c = Class.forName(className);) and a

NoSuchFieldException (line Field f = c.getField(fieldname);).
We can do this by either declare that all exception can be thrown or that these two specific exceptions can be thrown

Note: Valid Java programming language code must honor the Catch or Specify Requirement. This means that code that might throw certain exceptions must be enclosed by either of the following:

* A try statement that catches the exception. The try must provide a handler for the exception.
* A method that specifies that it can throw the exception. The method must provide a throws clause that lists the exception.
Code that fails to honor the Catch or Specify Requirement will not compile.

Reference: The Java Tutorials, The Catch or Specify Requirement

QUESTION NO: 70

You have been asked to create a ResourceBundle file to localize an application.

Which code example specifies valid keys of menu1 and menu2 with values of File Menu and View Menu?

A. <Key name = “menu1”>File Menu</Key>
<Key name = “menu2”>view Menu </Key>
B. <Key>menu1</key><value>File Menu </value>
<Key>menu2</key><value>View Menu </value>
C. menu1, File Menu, menu2, View Menu
D. menu1 = File Menu
menu2 = View Menu

Answer: D
Explanation: A properties file is a simple text file. You can create and maintain a properties file with just about any text editor.

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.

An example of the contents of a ResourceBundle file:

This is a comment

s1 = computer
s2 = disk
s3 = monitor
s4 = keyboard

Note that in the preceding file the comment lines begin with a pound sign (#). The other lines contain key-value pairs. The key is on the left side of the equal sign and the value is on the right. For instance, s2 is the key that corresponds to the value disk. The key is arbitrary. We could have called s2 something else, like msg5 or diskID. Once defined, however, the key should not change because it is referenced in the source code. The values may be changed. In fact, when your localizers create new properties files to accommodate additional languages, they will translate the values into various languages.

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

QUESTION NO: 71

Given:

```
class Fibonacci extends RecursiveTask<Integer> {

final int n;

Fibonacci (int n) { this.n = n }

Integer compute () {

if (n <= 1)

return n;

Fibonacci f1 = new Fibonacci (n – 1);

f1.fork;  // Line X

Fibonacci f2 = new Fibonacci (n – 2); // Line Y

return f2.compute() + f1.join;

}

}
```

Suppose that lines X and Y are transposed:

```
Fibonacci f2 = new Fibonacci (n – 2); // Line Y

f1.fork;  // Line X
```

What is the likely result?

A. The program produces the correct result, with similar performance to the original
B. The program produces the correct result, with performance degraded to the equivalent of being single-threaded.
C. The program produces an incorrect result
D. The program goes into an infinite loop
E. An exception is thrown at runtime
F. The program produces the correct result, the better performance than the original.

Answer: A
Explanation: The degree of parallelism is not changed. Functionality is the same.

QUESTION NO: 72

Given the incomplete pseudo-code for a fork/join framework applications:

```
submit (Data) {

if (Data.size < SMALL_ENOUGH) {

_____(Data);  // line X

}

else {

List<Data> x = _____(Data); // line Y

for(Data d: x

_____(d);  // line Z

}

}
```

And give the missing methods:

Process, submit, and splitInHalf

Which three insertions properly complete the pseudo-code?

A. Insert submit at line X
B. Insert splitHalf at line X
C. Insert process at line X
D. Insert process at line Y
E. Insert splitHalf at line Y
F. Insert process at line Z
G. Insert submit at line Z

Answer: C,E,G
Explanation: C: If Data.Size is "small enough" then process it directly.
E: Else we split it in half.
G: We use a recursive submit (of the splitted Data).

QUESTION NO: 73

The current working directory is named finance.

Which two code fragments allow you to write the salary.dat file if it does not exist under "finance\payroll"?

A. public static void setFileContent (String[] s) throws IOException {
path p=paths.get("payroll\\salary.dat");
File file=p.toAbsolutePath().toFile();
try (BufferWriter br = new BufferWriter (new FileWriter(File))) {
br.write (“experience new features of java”);
}
}
B. public static void setFileContent (String[] s) throws IOException {
path p=paths.get ("payroll\\salary.dat");
File file=p.toAbsolutePath(LinkOption.NOFOLLOW_LINKS).toFile();
try (BufferWriter br = new BufferWriter (new FileWriter(File)))
{ br.write (“experience new features of java”);
}
}
C. public static void setFileContent (String[] s) throws IOException
{ File file= new file ("payroll\\salary.dat").getCanonicalFile();
try (BufferWriter br = new BufferWriter (new FileWriter(File))) {
br.write (“experience new features of java”);

}
}
D. public static void setFileContent (String[] s) throws IOException
{ File file= new File ("payroll\\salary.dat").getCanonicalFile();
try (BufferWriter br = new BufferWriter (new FileWriter(File))) {
br.write (“experience new features of java”);
}
}
E. public static void setFileContent (String[] s) throws IOException
{ File file=new File ("payroll\\salary.dat").getAbsolutePath();
try (BufferWriter br = new BufferWriter (new FileWriter(File))) {
br.write (“experience new features of java”);
}
}

Answer: B,D
Explanation: The problem in this scenario is how to construct a system-dependent filename from the string "payroll\\salary.dat".

Regarding File-paths:
1- A file can have many relative paths.2- Canonical paths are absolute paths.3- An absolute path is not necessarily a canonical path! This holds trueespecially under Unix, which support symbolic links. Under Windows, anabsolute path is usually a canonical path.

B:The absolute path can include symbolic links. Here we ignore them with NOFOLLOW_LINKS option.
D: The File.getCanonicalFile Method creates a new instance of a File object representing the file located at the absolute path of the current File object. All '.' and '..' references will be resolved.

QUESTION NO: 74

Which statement is true about the DSYNC constant defined in standardopenoptions enums?

A. DSYNC keeps only the file content and not the metadata synchronized with the underlying storage device.
B. DSYNC keeps thefile (both content and metadata) synchronized with the underlying storage device.
C. DSYNC keeps only the file metadata and not the file content synchronized with the underlying storage device.
D. DSYNC keeps the file (both content and metadata) de-synchronized with copies in the und< storage device.

Answer: A
Explanation: DSYNC keeps the file content synchronized with the underlying storage device.

Note:SYNC keeps the file (both content and metadata) synchronized with the underlying storage device.

Note 2: These are from the java.nio.file.StandardOpenOption enum class.

QUESTION NO: 75

Given the following code fragment:

```
public static void getInfo() {

//insert code here
List fontCatalog = new ArrayList();

fontCatalog.add("Algerian");
fontCatalog.add("Cambria");
fontCatalog.add("Lucida Bright");
category.put("firstCategory",fontCatalog);

List entrySet = new ArrayList(category.entrySet());
Iterator it = entrySet.iterator();

while(it.hasNext())
{ System.out.println(it.next)
);
}
}
```

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

A. Map<String, List<String>> category = new HashMap<List> ();

B. Map<String, List<String>> category = new HashMap<<>,List<>>();
C. Map<String, List<String>> category = new HashMap<> ();
D. Map<String, List<String>> category = new HashMap <String, ArrayList<String>> ();
E. Map<String, List<String>> category = new HashMap<String, List<String>> ();
F. Map<String, List<String>> category = new HashMap<String, List <>> ();

Answer: C,E
Explanation: E: Redundant type arguments in new expressions. Use diamond operator instead.

QUESTION NO: 76

Given:

ConcurrentMap<String, String> partList = new ConcurrentHashMap<> ();

Which fragment puts a key/value pair in partList without the possibility of overwriting an existing key?

A. partList.put (key,"Blue Shirt");
B. partList.putAbsent(key,"Blu Shirt")
C. partList.putIfNotLocked (key,"Blue Shirt");
D. partList.putAtomic (key,"Blue Shirt");
E. if (!partlist.containsKey(key)) partList.put(key,"Blue Shirt");

Answer: E
Explanation: The containsKey method returns true if this map maps one or more keys to the specified value.
So this statement adds a new key if they key is not present.
Reference: Class ConcurrentHashMap<K,V>

QUESTION NO: 77

Which three objects must a vendor provide implementations in its JDBC driver?

A. Time
B. Date
C. Statement
D. RuleSet
E. Connection

F. SQLException
G. DriverManager

Answer: E,F,G
Explanation: E:When the method getConnection is called, the DriverManager will attempt to locate a suitable driver from amongst those loaded at initialization and those loaded explicitly using the same classloader as the current applet or application.
F: An SQLException is an exception that provides information on a database access error or other errors.
G: DriverManager is the basic service for managing a set of JDBC drivers.

Reference: The Java Tutorials,Class DriverManager

QUESTION NO: 78

Given:

```
public class MyGrades {

private final List<Integer> myGrades = new ArrayList<Integer>();


private final ReadWriteLock rwlock = new ReentrantReadWriteLock();


public void addGrade(Integer grade) {

/*

lock and modify

*/

}


public void averageGrades() {

// acquire _____lock Line **

double sum = 0;

int i = 0;

for (i = 0; i < myGrades.size(); i++) {

sum += myGrades.get(i);
```

```
}

// release _____lock Line ***

System.out.println("The average is: " + sum/(i+1));

}

}
```

Which pair's statements should you insert at line ** and line *** (respectively) to acquire and release the most appropriate lock?

A. rwlock.readLock().acquire();
rwlock.readLock().release();
B. rwlock.readLock().lock();
rwlock.readLock().unlock();
C. rwlock.getLock().acquire();
rwlock.getLock().release();
D. rwlock.getLock().lock();
rwlock.getLock().Unlock();
E. rwlock.WriteLock().acquire();
rwlock.writeLock().release();
F. rwlock.writeLock().lock();
rwlock.WriteLock().unlock();

Answer: B
Explanation: We need a read lock, not a write lock, we are just reading data, not writing/updating data.
To aquire and release the lock the method lock() and unlock are used.

Reference: Class ReentrantReadWriteLock

QUESTION NO: 79

Which two methods are defined in the FileStore class print disk space information?

A. getTotalSpace ()
B. getFreeSpace ()
C. getUsableSpace ()
D. getTotalCapacity ()
E. getUsed ()

Answer: A,C
Explanation: A: The getTotalSpace() method returns the size, in bytes, of the file store.
C: The getUsableSpace() method returns the number of bytes available to this Java virtual machine on the file store.

Reference: Class java.nio.file.FileStore

QUESTION NO: 80

Given the code fragment:

public static void main(String[] args) {

Path file = Paths.get("D:\\company\\report.txt");

try (SeekableByteChannel sbc = Files.newByteChannel(file,new OpenOption[]

What is the result if the report.txt file contains

Welcome to the world of Java?

A. The file contains the first five characters.
B. The file contains the first four characters.
C. The contents of the file remain unchanged.
D. A NonWritableChannelException is thrown at runtime.

Answer: C
Explanation: The truncate line will not change the file since the file size is less than 30.

Reference: Interface SeekableByteChannel

QUESTION NO: 81

What is the minimum SQL standard that a JDBC API implementation must support?

A. SQL-92
B. SQL 99

C. SQL-2003
D. JDBC 4.0

Answer: A
Explanation: JDBC sets minimum SQL conformance to the SQL92 entry level SQL standard. This gives guaranteed wide portability for applications designed to run on many platforms.

QUESTION NO: 82

Which two fragments can be combined into a statement that returns an ExecuteService instance?

A. Executors
B. Executor
C. ExecutorService
D. .getSingleThreadExecutor ();
E. .newSingleThreadExecutor ();
F. .createSingleThreadExecutor ();
G. .buildSingleThreadExecutor ();

Answer: A,E
Explanation: The Executors.newSingleThreadExecutor() method creates an Executor that uses a single worker thread operating off an unbounded queue.

Reference: java.util.concurrent.Executors

QUESTION NO: 83

Given the code fragment:

```
public class App {

public static void main (String [] args) {

path path = paths.get(“C\\educations\\institute\\student\\report.txt”);

system.out.printIn(“getName(0): %s”, path.getName(0));

system.out.prinIn(“subpath(0, 2): %s” path.subpath(0, 2));

}

}
```

What is the result?

A. getName (0): C:\
subpath(0, 2): C:\education\report.txt
B. getName (0): C:\
subpath(0, 2): education\institute
C. getName(0): education
subpath(0, 2: education\institute\student
D. getName(0): education
subpath(0, 2): education\institute
E. getName(0): report.txt
subpath (0, 2): institute\student

Answer: D
Explanation: The getName(int index) method returns a name element of this path as a Path object.

The subpath(int beginIndex, int endIndex) method returns a relative Path that is a subsequence of the name elements of this path.

Reference: java.nio.file.Path

QUESTION NO: 84

Given this code fragment:

```
try {
String query = "SELECT * FROM Item";
Statement  stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(query);
ResultSetMetaData rsmd = rs.getMetaData();
int rowCount = rsmd.getRowCount();
System.out.println ("Processing: " + rowCount + " rows.");
while (rs.next()) {
// Process each row
}
```

```
} catch (SQLException se)

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

}
```

Assume that the SQL query returns records. What is the result?

A. Compilation fails.
B. The program prints Error
C. An exception is thrown at runtime
D. The statement at line 16 execute

Answer: A
Explanation: There is no GetRowCount method in java.sql.ResultSetMetaData.
The following line will not compile:
int rowCount = rsmd.getRowCount();

Reference: java.sql.ResultSetMetaData

QUESTION NO: 85

In the Java SE7 API, which method is most commonly used by factories to instantiate objects?

A. new () **B.**
make () **C.**
create ()
D. getObject ()
E. getInstance ()
F. createObject ()

Answer: E
Explanation: For example: KeyFactory.getInstance, ObjectFactory. getObjectInstance.

QUESTION NO: 86

Which method or methods should you implement to create your own implementation of the java.nio.file.PathMatcher interface?

A. matches(Path)
B. matches(Path),
fails(Path)
C. matches(Path) ,
fails(Path),
enable(boolean)
D. matches(Path) ,
fails(Path) ,
setPreferred (String)

Answer: A
Explanation: The interface PathMatcher is an interface that is implemented by objects that perform match operations on paths.
The single method for this interface ismatches:
boolean matches(Path path)
Tells if given path matches this matcher's pattern.
Parameters:
path - the path to match
Returns:
true if, and only if, the path matches this matcher's pattern.

Reference: java.nio.file.PathMatcher

QUESTION NO: 87

Given:

```
public class MyGrades {

private final List<Integer> myGrades = new ArrayList<Integer>();

private final ReadWriteLock rwlock = new ReentrantReadWriteLock();

public void addGrade(Integer grade) {

// acquire _____lock

myGrades.add(grade);

// release _____lock

}
```

```
public void averageGrades() {

// acquire _____lock  Line **

double sum = 0;

int i = 0;

for (i = 0; i < myGrades.size(); i++) {

sum += myGrades.get(i);

}

// release _____lock Line ***

System.out.println("The average is: " + sum/(i+1));

}

}
```

Which pair's statements should you insert at lines ** and lines *** (respectively) to acquire and release the most appropriate lock?

A. rwlock.readLock().acquire();
rwlock.readLock().release();
B. rwlock.readLock().lock();
rwlock.readLock().unlock();
C. rwlock.getLock().acquire();
rwlock.getLock().release();
D. rwlock.getLock().lock();
rwlock.getLock().Unlock();
E. relock.WriteLock().acquire();
rwlock.writeLock().release();
F. rwlock.writeLock().lock();
rwlock.WriteLock().unlock();

Answer: B
Explanation: We need a read lock, not a write lock, we are just reading data, not writing/updating data.
To aquire and release the lock the method lock() and unlock are used.

Reference: Class ReentrantReadWriteLock

QUESTION NO: 88

Given:

```
public class StringApp {

public static void main(String[] args) {

String[] str="Java SE,java EE,Java ME,java FX".split(",");

int count=0;

for(int i=0;i<str.length;i++) {

switch(str[i]) { case

"Java SE":

count++; continue;

case "Java EE":

count++; break;

case "Java ME":

count++; break;

case "Java FX":

count++; break;

}

}

System.out.println("Total match found="+count);

}

}
```

What is the result?

A. 1
B. 2
C. 3
D. 4

Answer: B
Explanation: Java SE and Java ME will increase the count variable as it matches.
java EE and java FX does not match.

QUESTION NO: 89

Which two file attribute views support reading or updating the owner of a file?

A. AclFileAttributeView
B. DosFileAttributeView
C. FileOwnerAttributeView
D. FileStoreAttributeView
E. BasicFileAttributeView

Answer: A,C
Explanation: A: The AclFileAttributeView is a file attribute view that supports reading or updating a file's Access Control Lists (ACL) or file owner attributes.
C: The FileOwnerAttributeView.setOwner(UserPrincipal owner) method updates the file owner.

Reference:Interface AclFileAttributeView,Interface FileOwnerAttributeView

QUESTION NO: 90

Which two code blocks correctly initialize a Locale variable?

A. Locale loc1 = "UK";
B. Locale loc2 = Locale.get Instance ("ru");
C. Locale loc3 = Locale.getLocaleFactory("RU");
D. Locale loc4 = Locale.UK;
E. Locale loc5 = new Locale("ru", "RU");

Answer: D,E
Reference: The Java Tutorials,Creating a Locale