Question 18 of 80

Given the following code:

```
01. public class Test {
02.    public static void main(String[ ] args) {
03.       Employee myManager = new Manager( );
04.       Manager myExec = new Executive( );
05.    }
06. }
07. class Employee { }
08. class Manager extends Employee { }
09. class Executive extends Employee { }
```

What is the result?

Answers

- A. Compilation will fail due to an error on Line 04.
- B. Compilation will fail due to an error on Line 03.
- C. A runtime error will occur due to an error on Line 04.
- D. The program will compile and execute successfully.

Assistance

Show explanation | Show answer

The Executive class does not inherit from the Manager class. Therefore, it is invalid to assign an object of type Executive to a reference variable of type Manager. To correct the error on Line 04 and allow the code to compile, the Executive class would have to extend the Manager class on Line 09.

Question 19 of 80

You have two variables: a and b. You need to assign the value of variable b to variable a before decreasing the value of b by 1. Which code segment would achieve this goal?

Answers

- A. `a = b - 1;`
- B. `a = --b;`
- C. `a = b--;`
- D. `a-- = b;`

Assistance

Show explanation | Show answer

In an assignment statement where the unary operator -- is used as a postfix operator, the value on the right-hand side of the assignment is assigned to the variable on the left-hand side prior to the -- operator being applied. In this scenario, a = b--; will cause the original value of b to be assigned to the variable a before the value of b is decremented by 1.

Question 20 of 80

The following code will not compile:

```
01. public class Test {
02.     int myVar = 2;
03.     public static void main(String[ ] args) {
04.         System.out.println(myVar);
05.     }
06. }
```

What can you do to resolve the compilation error?

Answers

- A. Change Line 02 to the following:
  `public int myVar = 2;`
- B. Change Line 02 to the following:
  `String myVar = 2;`
- C. Change Line 02 to the following:
  `static int myVar = 2;`
- D. Change Line 04 to the following:
  `System.out.println(Test.myVar);`

Assistance

Show explanation | Show answer

A static variable belongs to the class rather than an instance of the class. A static method can only access variables from a class if the variables are declared as static, or if they are accessed from an instance of the class. In this case, the non-static compilation error is resolved by adding the static keyword to the myVar variable declaration.

Question 21 of 80

Given the code:

```
01. class Tester {
02.     public static void main(String[ ] args) {
03.         int i = 1;
04.         for ( ; ; ) {
05.             System.out.print(i + " ");
06.         }
07. }
```

What is the result when you compile and execute this program?

Answers

- A. Line 04 causes a compilation error.
- B. The program compiles but nothing is displayed.
- C. The program compiles successfully but throws an error at runtime due to the code on Line 04.
- D. The program infinitely outputs the number 1 to the console window.

Assistance

Show explanation | Show answer

The for statement in Java consists of three expressions separated by semicolons. The first expression is the initialization statement that initializes the loop. The second expression is the termination expression that determines when the loop should end. The third expression is the increment expression that either increments or decrements a value for controlling the looping. All three expressions of the for loop are optional.

When all three are omitted, an infinite loop is created. You can exit this version of an infinite for loop by including the breakstatement as one of the statements in the body of the loop. Since no break statement is used in the for loop on Line 04, the value of i, 1, will infinitely be printed to the console window.

Question 22 of 80

Given the code:

```
01. public class StringTest {
02.     String data;
03.
04.     public StringTest(String s) {
05.         data = s;
06.     }
07.
08.     public static void main(String[ ] args) {
09.         StringBuilder sb = new StringBuilder("10");
10.         System.out.println(new StringTest(sb));
11.     }
12. }
```

What is the result?

Answers

- A. StringTest@659e0bfd
- B. 10
- C. A runtime error will occur due to the code on Line 10.
- D. A compiler error will occur due to the code on Line 10.

Assistance

Show explanation | Show answer

There is no constructor in StringTest that takes a StringBuilder argument. A StringTest instance cannot be created by passing a StringTest constructor a StringBuilder object. Attempting to create a StringTest instance with a non-existent constructor will cause a compiler error.

Question 23 of 80

Given the code:

```
01. public class MyClass {
02.     // Insert code here
03.         System.out.println("Hello World!");
04.     }
05. }
```

Which two method declarations, inserted independently on Line 02, will allow the MyClass program to properly compile and execute?

Answers

- A. `public static int main(String[ ] args) {`
- B. `static public void main(String[ ] args) {`
- C. `public static void main(String[ ] args) {`
- D. `public static void main( ) {`
- E. `public static void main(String args) {`
- F. `public static void Main(String[ ] args) {`

Assistance

Show explanation | Show answer

All Java applications must contain a main method. This method is the entry point for the application and allows you to invoke the other methods needed by your program. The main method must be declared as public, static, and void. In addition, the method must be able to accept an array of type String as an argument. This allows you to pass values from the command line when executing your program.

The main method is the entry point and all Java applications must have one. In addition to containing an array of type String as an argument, the method must also be declared with the modifiers public, static, and void. While the convention is to order these modifiers as public static void, they can also be ordered as static public void.

Question 24 of 80

You are writing a program that performs numerous mathematical computations. While testing your program, you receive a "divide by zero" message and your program abruptly ends. In which exception category would this condition be included?

Answers

- A. Checked exception
- B. Runtime exception
- C. Mathematical exception
- D. Error

Assistance

Show explanation | Show answer

Runtime exceptions are the result of conditions present within the application that occur while the program is executing. Generally, the application cannot anticipate or recover from one of these conditions, and the application crashes. Runtime exceptions are usually an indication of a programming bug. This can include passing a null value to a constructor and logic errors, such as division by zero.

Question 25 of 80

Which two code segments, taken independently, correctly construct a two-dimensional array capable of storing strings?

Answers

- [ ] **A.**
```
String[ ][ ] firstName = new String[14];
```

- [ ] **B.**
```
String[ ] firstName;
firstName = new String[14][ ];
```

- [ ] **C.**
```
String[ ][ ] firstName = new String[14][ ];
```

- [ ] **D.**
```
String[ ][ ] firstName;
firstName = new String[14][ ];
```

Assistance

Show explanation | Show answer

One way to create a two-dimensional array is to create a variable that can hold a reference to a two-dimensional object. To do this, two square brackets must be placed to the immediate right the data type of the variable. You can then use an assignment statement to create a new two-dimensional array object and set the variable to point to the new object. Double square brackets must be used to declare a two-dimensional array.

The size of at least the first dimension must be defined when the array is initialized. As with one-dimensional arrays, you can create a two-dimensional array in a single line of code. This code combines the declaration and assignment of a two-dimensional array.

Question 26 of 80

Given the following code:

```
01. interface Organism {
02.     void consume( );
03.     void move( );
04. }
05. class Creature implements Organism {
06.     // Insert code here
07. }
```

Which code segment inserted on Line 06 will allow the code to correctly compile?

Answers

- A.
```
abstract void consume( );
abstract void move( );
```
- B.
```
public void consume( ) { }
public void move( ) { }
```
- C.
```
public abstract void consume( );
public abstract void move( );
```
- D.
```
void consume( ) { }
void move( ) { }
```

Assistance

Show explanation | Show answer

Methods declared in an interface are public and abstract by default. Therefore, when a class implements an interface, it must implement all of the public methods that exist in the interface. In this case, the Organism interface has two methods, consume and move. Since the Creature class implements Organism, it must implement all methods that exist within the Organism interface.

Question 27 of 80

Given the code:

```
01. public class Variables {
02.     public static void main(String[ ] args) {
03.         int myVariable = 5;
04.         Test myTest = new Test( );
05.     }
06. }
07. class Test {
08.     Variables myVar;
09.     char myChar;
10. }
```

Which two lines of code correctly declare a reference variable?

Answers

- A.
```
Test myTest = new Test( );
Variables myVar;
```
- B.
```
int myVariable = 5;
char myChar;
```
- C.
```
Variables myVar;
char myChar;
```
- D.
```
int myVariable = 5;
Test myTest = new Test( );
```

Assistance

Show explanation | Show answer

A reference variable is used to store a reference of an object in memory. Both the myTest and the myVar variables are correctly declared as reference variables. In this case, the variable myTest on Line 04 is declared as a reference variable and initialized while the variable myVaron Line 08 is declared as a reference variable but not initialized.

Question 28 of 80

Given the code:

```
public class NumberChange {

    public static void main(String [ ] Args) {
    int x = 10;
    NumberChange num = new NumberChange( );
    num.changeNumber(x);
    System.out.println("x is: " + x);
    }

    public void changeNumber(int x) {
    x = 15;
    }

}
```

What is the result?

Answers

- A. x is: 15
- B. The program does not compile.
- C. The program compiles but throws a runtime error.
- D. x is: 10

Assistance

Show explanation | Show answer

Question 29 of 80

Which two code segments, taken independently, will correctly create an array named nums containing the values 10, 20, and 30?

Answers

A.
```
int[ ] nums = new int[3];
nums[1] = 10;
nums[2] = 20;
nums[3] = 30;
```

B.
```
int[ ] nums = new int[3];
nums[0] = 10;
nums[1] = 20;
nums[2] = 30;
```

C.
```
int[3] nums = { 10, 20, 30 };
```

D.
```
int[ ] nums = { 10, 20, 30 };
```

Assistance

Show explanation | Show answer

One way to create an array in Java is to group the values within a pair of braces. This allows you to create the array and assign values to it in a single line of code. The int[ ] nums = { 10, 20, 30 }; code segment creates an array named nums of type int. It also assigns the value 10 to the first element in the array, 20 to the second element, and 30 to the third element. The size of the array created is determined by the number of values listed between the braces.

The int[ ] nums = new int[3]; code segment declares and creates an array of type int named nums with three elements. To assign a value to a position in an array the array name must first be referenced along with a set of square brackets that contain the index of the desired position.

The assignment operator can then be used to set a value for the specified position. Since arrays are a zero index-based structure, the first position in the array can be accessed using an index of 0. The following lines of code assign the values 10, 20, and 30 to position 1, 2, and 3 of the nums array.

```
nums[0] = 10;
nums[1] = 20;
nums[2] = 30;
```

Question 30 of 80

You are creating an application that takes a string of text and reverses the string. For example, if you input She sells sea shells the application will output sllehs aes slles ehS. What would be the easiest way to complete this task?

Answers

A.
```
String rhyme= "She sells sea shells";
StringBuilder sb = new
   StringBuilder(rhyme);
sb.reverse( );
```

B.
```
String rhyme= "She sells sea shells";
rhyme.reverse( );
```

C.
```
String rhyme= "She sells sea shells";
rhyme.intern( );
```

D.
```
String rhyme= "She sells sea shells";
StringBuilder sb = new
   StringBuilder(rhyme);
sb.lastIndexOf(rhyme.length( ));
```

Assistance

Show explanation | Show answer

The StringBuilder class allows you to take a string value and modify it. String objects are immutable. However, StringBuilder objects are treated as a variable length array that can change in length and value at any time using the methods of the StringBuilder class. In this scenario, calling the reverse method of the StringBuilder class will result in the value stored in the StringBuilder instance being completely reversed.

Question 31 of 80

Which two statements create valid reference variables?

Answers

- [ ] **A.** `int x = 1;`
- [ ] **B.**
```
MyClass mc1 = new MyClass( )
MyClass mc2 = mc1;
```
- [ ] **C.** `double x = 10.0;`
- [ ] **D.** `Object obj1 = new Object( );`

Assistance

Show explanation | Show answer

Java contains two data types: reference data types and primitive data types. Each variable needs to be associated with a declared data type. A reference variable is used to reference an object created and held in memory. Assigning a variable of type MyClass an object created with the new keyword will store the location in memory of the new object in the variable of type MyClass.

Assigning the value stored in a reference variable to another variable of type MyClass will cause the new variable to point to the same location in memory. Both variables that point to a location in memory are reference variables.

Assigning a variable of type Object an object created with the new keyword will store the location in memory of the new object in the variable of type Object. The variable of type Object that contains the location in memory of the new object is a reference variable.

Question 32 of 80

Given the following code:

```
01. A aClass = new A( );
02. A aClass2 = aClass;
03. // Insert code here
04.     System.out.println("objects are equal"); }
05. else{
06.     System.out.println("objects not equal"); }
```

Which code segment should you insert on Line 03 to test whether the values stored in aClass and aClass2 are equal?

Answers

- A. `if(aClass != aClass2) {`
- B. `if(aClass = aClass2) {`
- C. `if (aClass.equals(aClass2)) {`
- D. `if (aClass == aClass2) {`

Assistance

Show explanation | Show answer

The equals method is used to compare the values stored in two objects. It will test whether two objects are equal and return true when the values stored in two objects are the same.

Question 33 of 80

You are creating a Java program that will use a number of classes located in the java.io package. Which import statement should you use so that all of the classes in the java.io package are available in your program?

Answers

- A. `import java.io.*;`
- B. `import java.io.all;`
- C. `imports java.io.*;`
- D. `import java.io;`

Assistance

Show explanation | Show answer

The Java import statement allows you to use a class located in another package. When using the import statement, you specify the package name and the class name to which you want to access. If you want to use two or more classes from the same package, you can use the * wildcard character in the import statement. This will import all the classes in the specified package.

Question 34 of 80

In which Java loop construct are the statements in its body guaranteed to execute at least once?

Answers

- A. if
- B. for
- C. while
- D. do-while

Assistance

Show explanation | Show answer

The do-while statement is similar to the while statement. The main difference between the two is that the statements in the body of the doblock are guaranteed to execute a minimum of one time. This is because the exit condition for the loop is located at the end of the do-while statement and it is not executed until the statements in the do block have executed.

Question 35 of 80

Given the code:

```
01. interface One {
02.     void firstMethod( );
03. }
04. interface Two {
05.     abstract void secondMethod( );
06. }
07. interface Three extends One, Two {
08.     void thirdMethod( );
09. }
10. class Four implements Three {
11.     public void firstMethod( ) {
12.         ...
13.     }
14.     public void secondMethod( ) {
15.         ...
16.     }
17.     public void thirdMethod( ) {
18.         ...
19.     }
20. }
```

What is the result when you attempt to compile this code?

Answers

- A. The program will compile successfully.
- B. Compilation will fail due to an error on Line 05.
- C. Compilation will fail due to an error on Line 10.
- D. Compilation will fail due to an error on Line 07.

Assistance

Show explanation | Show answer

The given code is legal and will compile without any errors. In Java, a class can have at most one direct superclass. However, it is legal for interfaces to inherit from multiple interfaces. To extend multiple interfaces, your interface declaration will include a comma-delimited list of the interfaces that it extends. In this code, the class Four provides a method body for all methods included in the interfaces that it implements.

Question 36 of 80

You are creating an application that will accept input from the user and submit it to a database. While testing your application, you enter data of the correct type. However, the data is not in the correct format for the application or the database. Since your code does not verify the user input, the application attempts to submit the data to the database. A message displays indicating a problem with the data but the application recovers and continues executing. What would have caused this error message to be displayed?

Answers

- A. A runtime exception occurred
- B. An error occurred
- C. An unchecked exception occurred
- D. A checked exception occurred

Assistance

Show explanation | Show answer

Checked exceptions occur when there are conditions present during the execution of the application that cannot be controlled by the application itself. An example of a checked exception is a user entering an invalid or missing filename. A well-written program will catch and handle checked exceptions using a try-catch block. When a checked exception occurs, the try-catch block will handle the exception and notify the user for correction. This can prevent the abrupt termination of the program.

Question 37 of 80

Given the code:

```
01. public class MyClass {
02.     public static void main(String[ ] args) {
03.         for(int i = 0; i < args.length; i++) {
04.             System.out.print(args[i] + " ");
05.         }
06.     }
07. }
```

You have edited the PATH environment variable to include the bin directory of your JDK installation. You have also opened the command prompt and changed it to the directory where your source file is located. Which two command-line statements will properly compile and display Hello World to the console window?

Answers

- [ ] A. `java MyClass`
- [ ] B. `javac MyClass.java Hello World`
- [ ] C. `java MyClass Hello World`
- [ ] D. `javac MyClass.java`
- [ ] E. `java MyClass.class Hello World`
- [ ] F. `javac MyClass`

Assistance

Show explanation | Show answer

To compile a Java program, you use the javac.exe program. This program is located in the bin directory of your JDK installation. The correct syntax for the command is the keyword javac followed by the source file name and extension. When the source file is compiled, a file will be created with the same filename and the .class extension.

To run a compiled Java program, you use the java.exe program in the bin directory of your JDK installation. The correct syntax for the command is the keyword java followed by the filename without the .class extension. You can pass arguments to your program by including them after the filename, each separated by a space.

Question 38 of 80

Your program contains an array of type String named letters. You want to use an enhanced for statement to display the contents of the letters array to the console window. Which code segment should you use?

Answers

A.
```
foreach (String ltr in letters) {
    System.out.println(ltr);
}
```

B.
```
for (String[] ltr : letters) {
    System.out.println(ltr);
}
```

C.
```
for (String ltr : letters) {
    System.out.println(ltr);
}
```

D.
```
for (String ltr in letters) {
    System.out.println(ltr);
}
```

Assistance

Show explanation | Show answer

An enhanced for statement can be used to iterate through the elements in an array or collection. The type specified in the enhanced forstatement must match the type of the elements in the array or collection. In this scenario, the letters array contains elements of type String. Therefore, the specified type for the variable declared in the the enhanced for must also be declared as type String.

During each iteration of the enhanced for loop, the current value of the array or collection will be stored in the declared variable. This variable can then be passed to the System.out.println method to print the current value of the array or collection to the console window.

Question 39 of 80

Given the code:

```
import java.util.ArrayList;

public class MyClass {
   public static void main(String[ ] args) {
       ArrayList al = new ArrayList( );
       ...
   }
}
```

What is the initial capacity of the ArrayList al when this class is executed?

Answers

- A. Undefined
- B. Ten
- C. Zero
- D. Five

Assistance

Show explanation | Show answer

When you create an ArrayList using the no-argument constructor, Java creates an empty ArrayList that has the capacity to hold a total of ten elements. The size of the ArrayList will automatically grow as you add more elements than its current capacity. If you know beforehand how many elements your ArrayList will contain, you can use the version of the constructor that lets you specify the capacity when creating the ArrayList.

Question 40 of 80

Given the following class instances:

```
A aReference = new A();
B bReference = new B();
```

Which code segment correctly uses these class instances to write the value stored in the field named bString from the B class to the field named aString from the A class?

Answers

- A. `aReference.aString = bReference.bString;`
- B. `aString = bString;`
- C. `aReference.aString = new B().bString;`
- D. `A.aString = B.bString;`

Assistance

Show explanation | Show answer

When accessing a field from another class, you must first create an instance of the class. You can then use dot notation in the form of classInstance.fieldName to read the value stored in the bString field and write it to the aString field.

Question 41 of 80

Given the code:

```
01. public static void main(String[ ] args) {
02.     int x = 18;
03.     int y = 7;
04.     boolean a, b;
05.     a = false;
06.     if (x / 2 < y + 3)
07.         a = true;
08.     b = true;
09.     //Insert code here
10. }
```

Which two statements, inserted independently on Line 09, will evaluate to true?

Answers

- [ ] A. `if((x == 18) && (y < 7) ) { }`
- [ ] B. `if ( (x = x) && (y < x) ) { }`
- [ ] C. `if (a && b) { }`
- [ ] D. `boolean d = x > 13 && 7 <= y;`
- [ ] E.
```
if((((x / 2) > y) && a) &&
      (y * 5 <= 35 * 0.99))
   { }
```

Assistance

Show explanation | Show answer

The x variable is assigned a value of 18 on Line 02 and the y variable is assigned a value of 7 on Line 03. After substituting in the necessary values, the expression on Line 06 evaluates as follows:

```
18 / 2 < 7 + 3
9 < 10
TRUE
```

This evaluation will cause the code in the body of the if statement to be executed and a value of true to be assigned to the a variable. The b variable is then assigned a value of true on Line 08.

The logical operator && will only evaluate to true if both its left and right operands evaluate to true. Since a and b are both true, a && b will evaluate to true."

Substituting these values in the d = x > 13 && 7 <= y expression will evaluate as follows:

```
18 > 13 && 7 <= 7
true && true
```

The logical operator && will only evaluate to true if both its left and right operands evaluate to true. Since both sides of the && operator are true, the expression will evaluate to true.

Question 42 of 80

Given the code:

```
01. import java.util.Arrays;
02.
03. public class Test {
04.     public static void main(String[ ] args) {
05.         String[ ][ ] multiArray = new String[5][5];
06.         for(String[ ] a : multiArray) {
07.             Arrays.fill(a, "test");
08.         }
09.         for(String[ ] b : multiArray) {
10.             System.out.println( );
11.             // Insert code here
12.
13.             }
14.         }
15.     }
16. }
```

You want to print every element in all of the arrays that are stored in the multi-dimensional array multiArray. Which code segment should you insert on Line 11?

Answers

- A.
```
for(String c : b) {
    System.out.print(b + " ");
```
- B.
```
for(String[ ] c : multiArray) {
    System.out.print(c + " ");
```
- C.
```
for(String c : b) {
    System.out.print(c + " ");
```
- D.
```
for(String[ ] c : b) {
    System.out.print(c + " ");
```

Assistance

Show explanation | Show answer

In a multi-dimensional array, each element is another array. The contents of the first set of square brackets indicate how many arrays the multi-dimensional array will have, and the second set of square brackets indicates the size of the arrays. In this case, multiArray stores five arrays and each array stores five elements. To access all of the elements in each array, a double forloop is required.

Inserting an enhanced for loop on Line 11 that iterates through the contents of the current array can be used to print every element in multiArray.

Question 43 of 80

Given the code:

```
01. public class MyClass {
02.     public static void main(String[ ] args) {
03.         Double num = 2.0;
04.         MyClass myClass = new MyClass( );
05.         Test test = new Test( );
06.         myClass.number(test);
07.         System.out.println(test.num);
08.     }
09.     void number(Test test) {
10.         test.num = test.num * test.num;
11.     }
12. }
13. class Test {
14.     Double num = 3.0;
15. }
```

What is the result?

Answers

- A. 3
- B. 9
- C. The program does not compile.
- D. 2

Assistance

Show explanation | Show answer

When you pass an object reference as a parameter to a method, the method parameter will refer to the same object as the object reference. If the object is modified within the method, it will also be modified in the calling code. However, if the parameter that is passed to the method is a primitive value, then the calling code is not affected by changes to the variable that occur within the method.

In this case, when the number method is called on Line 06, an object reference named test is passed as a parameter. Therefore, the modifications that are made to the object reference test will remain modified in the code that called the method. The print statement on Line 07 prints the modified value of the num variable. This results in 9.0 being printed to the console window.

Question 44 of 80

Given the code:

```
01. public class Animal {
02.     static int height = 10;
03.     public static void main(String[ ] args) {
04.         Mammal gorilla = new Mammal("gorilla");
05.         height = 30;
06.         System.out.println(gorilla.name);
07.         System.out.println(gorilla.height);
08.     }
09. }
10. class Mammal extends Animal {
11.     String name;
12.     Mammal(String mammalName) {
13.         name = mammalName;
14.     }
15. }
```

What is the result?

Answers

- A. Compilation will fail due to an error on Line 07.
- B. Compilation will fail due to an error on Line 05.
- C. gorilla
  10
- D. gorilla
  30

Assistance

Show explanation | Show answer

A class can inherit the fields and methods of another class by using the extends keyword in the class declaration. In this case, a new instance of the Mammal class is created on Line 04 using the constructor that takes a string as a parameter. The gorilla string passed to the Mammal constructor on Line 04 will assign this value to the name variable of the Mammal class. The print statement on Line 06 will access the name variable located in the Mammal class and will print gorilla to the console window.

Since the Mammal class extends the Animal class, the Mammal class inherits the fields and methods of the Animal class. The heightvariable is a static variable and therefore belongs to the class. When the value of the height variable is changed on Line 05 to a value of 30, all instances that access the height variable will return a value of 30. The print statement on Line 07 accesses the height variable through the gorilla instance of the Mammal class. It then prints the current value of the height variable that is located in the Animal class.

Question 45 of 80

Given the code:

```
01. import java.io.*;
02.
03. class MyClass {
04.
05.     // Insert code here
06.     {
07.          if (str == null)
08.              throw new IOException( );
09.
10.          ...
11.          return "Done";
12.     }
13.
14.     public static void main(String[ ] args) {
15.         String s = null;
16.         try {
17.             String tester = getStr(s);
18.             System.out.println("The value of tester is: " + tester);
19.         } catch (IOException ex) {
20.              System.out.println("IOException caught");
21.         }
22.     }
24.
25. }
```

Which getStr method declaration inserted on Line 05 will allow the code to compile and display IOException caught to the console?

Answers

- A. `public static String getStr(String str) throws Exception`
- B. `public static String getStr(String str) throws IOException`
- C. `public static String getStr(String str)`
- D. `public static String getStr(String str) throw IOException`

Assistance

Show explanation | Show answer

You have two options for catching exceptions in a method: handle the error in the method where it occurs or throw the exception back to the caller to handle it. In this scenario, the getStr method throws an IOException back to the calling method to handle the error.
Since IOException is a checked exception and getStr does not handle IOExceptions, its method declaration must specify that it can throw this type of exception.

Question 46 of 80

Given the code:

```
01. public class TestAnimals {
02.     public static void main(String[ ] args) {
03.         Animal animal1 = new Dog( );
04.         Animal animal2 = new Monkey( );
05.         Animal animal3 = new Animal( );
06.         animal1.walk( );
07.         animal2.walk( );
08.         animal3.walk( );
09.     }
10. }
11. class Animal {
12.     void walk( ) {
13.         System.out.print("walking ");
14.     }
15. }
16. class Dog extends Animal {
17.     void walk( ) {
18.         super.walk( );
19.         System.out.println("a Dog");
20.     }
21. }
22. class Monkey extends Animal {
23.     void walk( ) {
24.         System.out.println("a Monkey");
25.     }
26. }
```

What is the result?

Answers

- A. The program will not compile.
- B. walking a Dog
  walking a Monkey
  walking
- C. walking
  a Dog
  a Monkey
  walking
- D. walking a Dog
  a Monkey
  walking

Assistance

Show explanation | Show answer

Polymorphism refers to when a subclass has some of the functionality of the parent class, but also implements some of its own features.

In this case, the Dog class and the Monkey class both implement their own walk method. The walk method of the Dog class is the first to be called on Line 06. This version of the walk method uses the super keyword to call the walk method of its parent class, which will print walking to the console window. Then, the walk method of the Dog class will print a Dog followed by a new line to the console window. The program then exits the method and returns to the main method where it calls the walk method of the Monkey class on Line 07. This version of the walk method will print a Monkey and a new line to the console window. The program will then return to the main method where it will call the walk method of the Animal class on Line 08. This version of the walk method will print walking to the console window.

Question 47 of 80

Which statement about the do-while loop in Java is true?

Answers

- **A.** Statements in the loop may never execute.
- **B.** The boolean expression is evaluated at the beginning of each iteration through the loop.
- **C.** Statements in the loop are guaranteed to execute at least once.
- **D.** The boolean expression is only evaluated at the end of the first iteration through the loop.

Assistance

Show explanation | Show answer

The do-while statement is similar to the while statement. Both statements allow a block of statements to execute based on the result of a boolean expression. The difference between the two is the do-while statement evaluates the boolean expression at the end of the loop instead of the beginning of the loop. This means that the block of statements in a do-while loop will always execute at least one time.

At the end of the loop, the boolean expression is then evaluated. If it evaluates to true, the loop will execute again. Otherwise, the loop will exit and execution will continue at the first statement after the do-while statement.

Question 48 of 80

Given the code:

```
01. public class Looping {
02.     public static void main(String[ ] args) {
03.         int a = 0;
04.         int b = 10;
05.         for (int c = 0; b > c; b--) {
06.             c++;
07.         }
08.         System.out.print(a + " " + b + " " + c);
09.     }
10. }
```

What is the result?

Answers

- **A.** 0 5 6
- **B.** 0 6 6
- **C.** 0 6 5
- **D.** A compilation error occurs.

Assistance

Show explanation | Show answer

The variable c is declared within the for statement and is only accessible within that block of code. Since Line 08 is outside the block of code where the variable c is declared, c cannot be accessed. The variable c is said to be out of scope at Line 08. Attempting to access the variable c on Line 08 will cause a compiler error.

Question 49 of 80

You are creating a program that contains a class named Employee and a class named Employer. You have created the following instance of the Employee class inside of the Employer class:

```
Employee emp = new Employee();
```

Which two code segments will allow you to read the value of a member variable named empName that is located in the Employee class from within the Employer class?

Answers

- [ ] **A.** `emp.empName;`
- [ ] **B.** `Employee.empName;`
- [ ] **C.** `empName;`
- [ ] **D.** `new Employee().empName;`

Assistance

Show explanation | Show answer

To access a field from a class, you first need to create an instance of the class. You can then use dot notation in the form of instanceName.fieldName to access the field. The emp.empName; code segment correctly accesses the empName field through the emp Employee instance.

You do not have to create a reference variable to store an instance of a class. However, you will have to use a new instance of the class using the new keyword every time you need to access the field if you do not store the instance in a reference variable.

Question 50 of 80

Given the following code:

```
01. int a = 0;
02. try {
03.     a = 5 / 0;
04.     System.out.println("Inside try: a = " + a);
05. }
06. catch(ArithmeticException ex) {
07.     System.out.print("Cannot divide by zero");
08.     System.out.println( );
09. }
10. System.out.println("Outside try: a = " + a);
```

What is printed to the console window?

Answers

- **A.** Inside try: a = 0
  Outside try: a = 0
- **B.** Cannot divide by zero
  Outside try: a = 0
- **C.** Cannot divide by zero
- **D.** Inside try: a = 0
  Cannot divide by zero
  Outside try: a = 0

Assistance

Show explanation | Show answer

Code that could cause an exception should be placed in the try block of a try-catch statement. Code to handle errors that may occur in the try block can be placed inside the catch block. When an error occurs, the program execution will jump to the catch block on Line 06. In this case, an error occurs in the try block when attempting to divide by 0 on Line 03.

Program control then jumps to the catch block and prints Cannot divide by zero to the console window. The program then exits the try-catch block and prints Outside try: a = 0 to the console window.

Question 51 of 80

Which two statements are true concerning the equals method and the == operator?

Answers

- [ ] **A.** The == operator is used to check whether two object references refer to the same object.
- [ ] **B.** The equals method is used to check whether two object references refer to the same object.
- [ ] **C.** The equals method is used to check whether the values stored in two objects are equal.
- [ ] **D.** The == operator is used to check whether the values stored in two objects are equal.

Assistance

Show explanation | Show answer

When comparing objects, it is important to note that the == operator and the equals method have distinct behaviors. The == operator is used to compare object references and will return true only when two references refer to the same object in memory.
The equals method is used to compare the values stored in two separate objects.

Question 52 of 80

You are creating a program that will allow users to set memos for a specific time and day in a calendar. You need to obtain the current date, along with the time. Which code segment should you use?

Answers

A.
```
LocalDateTime date = new LocalDateTime( ).
    getHour( );
```

B.
```
LocalDateTime date = new LocalDateTime( );
```

C.
```
LocalDateTime date = LocalDateTime.
    getHour( );
```

D.
```
LocalDateTime date = LocalDateTime.now( );
```

Assistance

Show explanation | Show answer

The LocalDateTime class can be used to obtain the current date and time. However, this class does not contain any public constructors. An instance of the LocalDateTime class can be created with the static now method of the LocalDateTime class.

Question 53 of 80

Given the code:

```
01. public class NumberChange {
02.
03.     public static void main(String [ ] Args) {
04.         NumberChange num = new NumberChange( );
05.         Integer[ ] x = new Integer[3];
06.         num.changeNumber(x);
07.         for (int y = 0; y < x.length; y++) {
08.             System.out.print(x[y] + " ");
09.         }
10.     }
11.
12.     public void changeNumber(Integer[ ] x) {
13.         for (int y = 0; y < x.length; y++) {
14.             x[y] = new Integer(y);
15.         }
16.     }
17.
18. }
```

What is the result?

Answers

- **A.** 0 0 0
- **B.** null null null
- **C.** 0 1 2
- **D.** 1 2 3

Assistance

Show explanation | Show answer

When you pass an object as a method parameter, the reference variable to the object is passed by reference. This means that the reference variable that is passed will still refer to the same object as it did when the method finishes and execution is returned to the calling line of code. Since the same object is being referenced before and after the method call, any changes that are made to the object inside the method will be reflected in the object when the method finishes.

In this code, Line 05 creates the reference variable x referring to an array of three Integer objects initialized to null. Line 06 calls the changeNumber method and passes the reference variable x as a parameter. The changeNumber method assigns three Integer objects to the array (0, 1, and 2). Since the variable x was passed by reference, the changes made to the array will be present when the changeNumber method ends. Therefore, 0 1 2 will be displayed as the result of the for loop on Line 07.

Question 54 of 80

Given the code:

```
01. class Test {
02.     public static int num = 0;
03. }
04. class TestDemo2 {
05.     public static void main(String[ ] args) {
06.         Test a = new Test( );
07.         Test b = new Test( );
08.         a.num += 1;
09.         b.num += a.num;
10.         System.out.println("a.num: " + a.num);
11.         System.out.println("b.num: " + b.num);
12.     }
13. }
```

What is the result?

Answers

- **A.** `a.num: 1`
  `b.num: 1`
- **B.** The program runs with no output.
- **C.** `a.num: 2`
  `b.num: 2`
- **D.** `a.num: 1`
  `b.num: 2`
- **E.** The program fails to compile due to an error on Line 02.

Assistance

Show explanation | Show answer

When you apply the static modifier to a field in a class, it is known as a static field or class variable. A class variable is a single variable that is shared by all instances of the class. If one instance of the class makes a change to the value of a class variable, all instances of the class will see that change.

On Line 02, num is declared as a class variable and initialized to a value of 0. Therefore, both a.num and b.num refer to the same value in memory. Line 08 increments num to a value of 1. Line 09 adds num to itself for a final value of 2. Since both a.num and b.num share the numvariable, both print statements will display the value 2.

Question 55 of 80

You have created the following code to test wrapper classes.

```
01. int myNum = new Integer(5);
02. System.out.println(myNum.doubleValue( ));
```

What is the result?

Answers

- **A.** 5
- **B.** A compiler error will occur due to the code on Line 02.
- **C.** A runtime error will occur due to the code on Line 02.
- **D.** A compiler error will occur due to the code on Line 01.

Assistance

Show explanation | Show answer

Primitive variables are variables that are used to store values belonging to one of the primitive type values. Each primitive type has a wrapper class that is used during autoboxing and autounboxing. While each wrapper class contains methods that can be executed using the value stored in the object, these methods cannot be accessed through a primitive variable. Attempting to do so will cause a compiler error.

Since the code on Line 02 attempts to access the doubleValue method of the Integer class, a compiler error will occur.

Question 56 of 80

What is an advantage of Java?

Answers

- A. Java code can be compiled without installing additional software.
- B. Java bytecode can be executed on any platform where a JVM is available.
- C. The Java Virtual machine (JVM) is platform independent.
- D. A .java file can be executed on any platform.

Assistance

Show explanation | Show answer

The JVM is used to execute the Java bytecode that is created when a Java program is compiled. Although there is a JVM available for most operating systems, the JVM to use depends on the operating system being used. While Java bytecode is platform independent and can be executed on any platform, the JVM is not.

Question 57 of 80

Given the code:

```
switch (num) {
    case 0:
        System.out.print("foo");
        break;
    case 1:
        System.out.print("bar");
        break;
    case 2:
        System.out.print("foofoo");
    case 3:
        System.out.print("barbar");
    case 4:
        System.out.print("foobar");
        break;
    case 5:
        System.out.print("barfoo");
        break;
    default:
        System.out.print("Finished");
}
```

What will be displayed to the console window if the variable num has a value of 2?

Answers

- A. `foofoobarbarfoobar`
- B. `foofooFinished`
- C. A runtime error occurs.
- D. `foofoo`

Assistance

Show explanation | Show answer

The switch statement in Java uses the value of an expression to determine which block of code will execute. If the value of the expression matches the value for one of the case statements, that block of code will execute. Execution will continue in that block until a breakstatement or the end of the switch statement is reached. If the block of code does not have a break statement, the statements in the next case will also execute.

In this case, the variable num has a value of 2. Since this matches one of the cases, the print statement in case 2 will execute and display foofoo to the console window. Since case 2 does not have a break statement, the print statement in case 3 will execute. This results in barbar being displayed to the console window. Again, with no break statement present, case 4 will execute and display foobar to the console window. The break statement in case 4 causes the execution to jump to the end of the switch statement.

Question 58 of 80

Given the code:

```
01. public class CmdLineInput {
02.     public static void main(String[ ] args) {
03.         int i = 0;
04.         do {
05.             if (args.length == 0) {
06.                 System.out.println("No input");
07.                 break;
08.             }
09.             System.out.print(args[i] + " ");
10.             i++;
11.         } while (i < args.length);
12.         System.out.print("Finished");
13.     }
14. }
```

You compile the code and enter the following command at the command prompt:

```
java CmdLineInput
```

What is the result?

Answers

- A. Line 11 causes a runtime error.
- B. No input
  Finished
- C. No input
- D. Finished

Assistance

Show explanation | Show answer

The break statement in Java can be used to terminate a switch statement. The break statement can also be used to terminate looping structures including for, while, and do-while loops.

The command used to execute this program does not include any arguments. Therefore, the if statement on Line 05 will evaluate to true. Line 06 will display No input and then the break statement on Line 07 will execute. The break statement causes control to jump out of the do-while loop to Line 12. The print statement then displays Finished and the program terminates.

Question 59 of 80

Given the code:

```
01. class A {
02.     public void getMessage( ) {
03.         System.out.println("In class A");
04.     }
05. }
06.
07. class B extends A {
08.     public void getMessage( ) {
09.         System.out.println("In class B");
10.     }
11. }
12.
13. class Tester {
14.     public static void main(String[ ] args) {
15.         A a1 = new A( );
16.         B b1 = new B( );
17.         a1 = b1;
18.         B b2 = a1;
19.         a1.getMessage( );
20.         b2.getMessage( );
21.     }
22. }
```

What is the result?

Answers

- A. Compilation fails due to an error on Line 18.
- B. In class B
  In class A
- C. Compilation fails due to error on Line 17.
- D. The program compiles but throws a runtime error.
- E. In class A
  In class B

Assistance

Show explanation | Show answer

A reference variable of one type can refer to an object of another type when the reference variable refers to an object that is a subclass of its own type. In this code, class B is a subclass of class A. Therefore, it is possible to assign an object of type B to a reference variable of type A. However, the reverse is not true. Line 18 is attempting to assign an object of type A to a reference variable of type B. Since this is illegal, an incompatible types compilation error is thrown.

Question 60 of 80

Given the code:

```
01. DateTimeFormatter formatter =
02.     DateTimeFormatter.ofPattern("DD MM YYYY");
03. LocalDate date = LocalDate.of(2015, 12, 20);
04.
05. System.out.println(date.format(formatter));
```

What is the result?

Answers

- A. 354 12 2015
- B. 20-12-2015
- C. 20 12 2015
- D. DateTimeException

Assistance

Show explanation | Show answer

The ofPattern method of the DateTimeFormatter class can be used to generate a DateTimeFormatter that uses a specific pattern. The D, M, and Y symbols are reserved pattern letters that can be used to define a pattern. The D symbol represents the day of year as a number, the M symbol represents the month of year as text or a number, and the Y symbol represents a week based year as a year. The number of times a symbol is used in a pattern will determine how a date will be formatted. The "DD MM YYYY" pattern will return two numbers for the day of year, two numbers for the month of year, and the week based year using four values.

The format method of the LocalDate class can be used to obtain a formatted version of the date as a String object. This method must be passed a valid DateTimeFormatter to specify how the date should be formatted. In this case, the day of year for December 20, 2015 would be 354. Since the DateTimeFormatter defined on Line 02 only allows two numbers for the day of year, 354 cannot be formatted correctly. This will cause a DateTimeException runtime error that will prevent the code from executing successfully.

Question 61 of 80

Given the code:

```
import java.io.*;
public class Test {
    public static void main(String[ ] args) {
        Console cons = System.console( );
        try {
            System.out.println("Please enter your age.");
            int age = Integer.parseInt(cons.readLine( ));
            System.out.println("You are " + age + " years old.");
        }
        catch(Exception e) {
            System.out.println("Error: A number is required.");
        }
        finally {
            System.out.println("Exit");
        }
    }
}
```

When prompted for input, the user enters Ted in the command window. What is displayed to the console window?

Answers

- A. Please enter your age.
  Ted
  You are Ted years old
- B. Please enter your age.
  Ted
  Error: A number is required.
- C. Please enter your age.
  Ted
  Error: A number is required.
  Exit
- D. Please enter your age.
  Ted
  You are Ted years old.
  Exit

Assistance

Show explanation | Show answer

Code that could cause an exception should be placed in the try block of a try-catch statement. Code to handle errors that occur can be placed inside the catch block. When an error occurs, the program execution will jump to the code in the catch block. When a finallyblock is added to a try-catch block, it will execute even if no error is thrown in the try block.

In this case, the program enters the try block and prints Please enter your age. to the console window. The user then inputs Ted via the console window. The program takes the input and attempts to parse it as an int. Since Ted cannot be parsed as an int, an error is thrown and the program execution jumps to the catch block. Inside the catch block, Error: A number is required. is printed to the console window. Next, the program exits the catch block and enters the finally block, where Exit is printed to the console window.

Question 62 of 80

Given the code:

```
01. public class Squares {
02.     public static void main(String[ ] args) {
03.         int x = 2;
04.         System.out.println("x squared = " + squareNumber(x));
05.         System.out.println("x = " + x);
06.     }
07.
08.     public static int squareNumber(int x) {
09.         x = x * x;
10.         return x;
11.     }
12. }
```

What is the result?

Answers

- A. x squared = 4
  x = 4
- B. x squared = 4
  x = 2
- C. A runtime error occurs.
- D. A compilation error occurs.

Assistance

Show explanation | Show answer

The print statement on Line 04 will pass the value of x from Line 02 to the squareNumber method. This method squares the value and returns 4. This is then displayed to the console window.

A variable declared within a method is known as a local variable and is accessible only from within that method. Therefore, any reference to a variable named x within the main method will refer to the local variable x that is declared on Line 03. As a result, the print statement on Line 05 is referring to the variable x from Line 03 and will display the value 2 to the console window.

However, the print statement on Line 05 will not display the value of 4. The scope for the variable x declared on Line 03 is in the mainmethod only, while the scope for the variable x declared as a parameter on Line 08 is in the squareNumber method only. When the print statement on Line 05 is executed, the variable x that is being referenced will be the x that is in scope for the main method. Since this value never changes in the scope of the main method, x = 4 cannot be printed to the console window.

Question 63 of 80

Given the following code:

```
01. public static void main(String[ ] args) {
02.     int[ ] numbers = {1, 5, 8, 9, 22, 10};
03.     myClass calc = new myClass( );
04.     System.out.print(calc.sum(numbers));
05. }
06. void sum(int[ ] numbers) {
07.     return true;
08. }
```

What is the result?

Answers

- A. TRUE
- B. Compilation will fail due to an error on Line 07.
- C. Compilation will fail due to an error on Line 06.
- D. 55

Assistance

Show explanation | Show answer

You can use a return statement with a method that has been declared void. However, you cannot specify a value with the return keyword when the method is declared with the void return type, as the void return type does not return a value.

In this case, the sum method is declared with a return type of void on Line 06. Since Line 07 contains a return statement that returns a value of true in the body of the sum method, a compilation error will occur.

Question 64 of 80

Which two Java statements evaluate their boolean expression at the beginning of each iteration of the loop?

Answers

- [ ] A. while
- [ ] B. for
- [ ] C. if
- [ ] D. do-while

Assistance

Show explanation | Show answer

The while statement can be used to repeatedly execute a block of statements. When the while statement is executed, it first evaluates the exit condition for the loop. If the boolean expression in the while statement evaluates to true, the block of statements in the loop are executed. When the loop finishes, control jumps to the start of the loop where the boolean expression is again evaluated.
The booleanexpression will continue to be evaluated at the beginning of each iteration until it evaluates to false. When this occurs, the while statement is exited and execution continues on the first line of code following the while statement.

Like the while statement, the for statement allows you to execute a block of statements repeatedly until a specified condition is no longer met. The for statement is useful in situations where you know how many times you want to repeat a block of statements. The forstatement uses a boolean expression to determine when the loop should be exited. This boolean expression is evaluated at the beginning of each iteration of the loop. If the expression evaluates to true, the statements in the loop are executed; otherwise, the for statement is exited. Execution then continues on the first line of code following the for statement.

Question 65 of 80

From which class do all Java exception classes derive?

Answers

- A. `java.lang.Error`
- B. `java.lang.Exception`
- C. `java.io.IOException`
- D. `java.lang.RuntimeException`

Assistance

Show explanation | Show answer

All exception classes in Java derive from the Exception class, located in the java.lang package. In addition to catching more specific exceptions, most try-catch structures include a catch block to catch Exception as well. Since of the way a try-catch structure works, the catch block that catches Exception must appear last. By doing this, any exceptions that are not caught by the other catch blocks can be caught and handled by the catch block for Exception.

Question 66 of 80

Given the code:

```
01. interface Playable {
02.     public void play( );
03. }
04.
05. class Instrument implements Playable {
06.     public void play( ) {
07.         System.out.println("music");
08.     }
09. }
10.
11. class StringedInstrument extends Instrument {
12.     public void play( ) {
13.         System.out.println("stringed instrument");
14.     }
15. }
16.
17. class Violin extends StringedInstrument {
18.     public void play( ) {
19.         System.out.println("pluck");
20.     }
21. }
22.
23. class Drum extends Instrument {
24.     public void play( ) {
25.         System.out.println("beat");
26.     }
27. }
28.
29. class PlayMusic {
30.     public static void main(String[ ] args) {
31.         Playable[ ] myInstruments = {new Instrument( ), new Violin( ), new Drum( )};
32.         for (Playable x: myInstruments) {
33.             x.play( );
34.         }
35.     }
36. }
```

What is the result?

Answers

- A. music
  stringed instrument
  beat
- B. Compilation fails due to an error on Line 31.
- C. music
  pluck
  beat
- D. music
  music
  music
- E. stringed instrument
  music
  pluck

Assistance

Show explanation | Show answer

Java allows a reference variable of one type to refer to objects of another type. However, for this behavior to occur, the reference variable can only refer to objects that are subtypes of its own type. In this code, the Instrument class implements the Playable interface and is inherited by the StringedInstrument, Violin, and Drum classes. Therefore, a reference of type Playable is able to refer to any of these classes.

At runtime, the JVM is able to determine which object type the reference refers to and calls the most appropriate instance method. This produces the output of music, pluck, and beat, on three separate lines.

Question 67 of 80

You are creating a program to test students on their knowledge of the solar system. Your program includes the code:

```
switch (planet) {
    case "Mercury":
    case "Venus":
    case "Earth":
    case "Mars":
        System.out.println("This is an inner planet.");
        break;
    case "Jupiter":
    case "Saturn":
    case "Uranus":
    case "Neptune":
        System.out.println("This is an outer planet.");
        break;
    default:
        System.out.println("Not a planet in our solar system.");
}
```

What is the result if the value of planet is set to "Earth"?

Answers

- A. This is an outer planet.
- B. This is an inner planet.
  Not a planet in our solar system.
- C. This is an inner planet.
  This is an outer planet.
- D. Not a planet in our solar system.
- E. This is an inner planet.

Assistance

Show explanation | Show answer

The switch statement in Java uses the value of an expression to determine which block of code will execute. If the value of the expression matches the value for one of the case statements, that block of code will execute. Execution will continue in that block until a breakstatement or the end of the switch statement is reached. If the block of code does not have a break statement, the statements in the next case will also execute.

In this code, the expression of the switch statement evaluates to "Earth". Since this matches one of the case values, execution jumps to that case. Because the "Earth" case does not include a break statement, execution continues into the "Mars" case. This results in This is an inner planet. being displayed to the console window. Since the next line of code is a break statement, execution will jump to the first statement following the switch.

Question 68 of 80

Given the following code segment:

```
01. ArrayList myArrayList = new ArrayList( );
02. myArrayList.add(1);
03. myArrayList.add(2);
04. myArrayList.add(3);
05. myArrayList.remove(1);
06.
07. for(Object b: myArrayList) {
08.    System.out.print(b + " ");
09. }
```

What is the output?

Answers

- **A.** 1 2 3
- **B.** 1 3
- **C.** 2 3
- **D.** Compilation will fail due to an error on Line 05.

Assistance

Show explanation | Show answer

The remove method on Line 05 can take an index as a parameter to remove the element at the location indicated by the index. In this case, the remove method takes 1 as a parameter. Since an ArrayList is zero index-based, the remove method will remove the element located at the second position of the ArrayList. Therefore, 1 3 will be printed to the console window.

Question 69 of 80

While some components of a class are optional, every class must have at least one of which item?

Answers

- **A.** A constructor
- **B.** A main method
- **C.** A return type
- **D.** A field

Assistance

Show explanation | Show answer

Every class that you create must have a minimum of one constructor. A constructor is a special method that is used to create an instance of the class in which it is defined. Unlike regular methods, a constructor does not have a return type and it must have the same name as the class. If you do not explicitly define a constructor, the Java compiler will add a no-argument, default constructor for you. This ensures that all classes meet the minimum requirement of one constructor.

Question 70 of 80

You are creating a program that utilizes the following String object:

```
String s = "Hello World";
```

Which code segment can you use to print World to the console window?

Answers

A. `System.out.println(s.indexOf(s.length( )));`

B. `System.out.println(s.indexOf(5));`

C.
```
System.out.println(s.substring(5,
    s.length( )));
```

D.
```
System.out.println(s.substring(6,
    s.length( )));
```

Assistance

Show explanation | Show answer

The substring method of the String class can be used to obtain a substring of a String object. The first argument of this method is the starting index of the substring in the original string. The starting index for a substring is inclusive. The second argument of the substringmethod is the ending index of the substring in the original string. The ending index of a substring is exclusive.

The W character in Hello World has an index of 6 and the d character has an index of 10. Since the ending index of the substring method is exclusive and the starting index is inclusive, passing substring 6, and the length of Hello World, in that order, will return World.

Question 71 of 80

Given the code:

```
int x = 5, y = 10;
do {
   --x;
   System.out.print(x + " ");
   y++;
} while (y < 12);
```

What is the result?

Answers

- **A.** 5 4
- **B.** 4 3
- **C.** A runtime error
- **D.** A compilation error

Assistance

Show explanation | Show answer

The do-while loop construct in Java guarantees that the body of the loop will execute at least once. This is because the exit condition for the loop is not checked until the end of the loop. In this code, the variable x is first reduced by 1 to a value of 4 and then printed to the console window. The y variable is then increased by 1 to a value of 11. The while condition is then checked and because (11 < 12)evaluates to true, execution returns to the start of the do-while loop.

The variable x is again reduced by 1, resulting in 3 being printed. The y variable is then increased by 1 to 12. Because this causes the while condition to evaluate to false, the do-while loop is exited.

Question 72 of 80

Given the code:

```
01. class Fish {
02.     int numFins;
03.     int numGills;
04.
05.     public Fish(int nFins, int nGills) {
06.         numFins = nFins;
07.         numGills = nGills;
08.     }
09.     public static void main(String[ ] args) {
10.         AngelFish angelFish = new AngelFish(4, 3, "Black, Silver");
11.         AngelFish angelFish2 = new AngelFish( );
12.     }
13. }
14. class AngelFish extends Fish {
15.     String colors;
16.     public AngelFish(int fins, int gills, String fColors) {
17.         super(fins, gills);
18.         colors = fColors;
19.     }
20.     public AngelFish( ) {
21.
22.         colors = "Silver";
23.
24.     }
25. }
```

The code currently does not compile. Which two code segments, inserted independently, will allow the code to compile correctly?

Answers

- [ ] **A.** Insert on Line 23:
```
super(2, 3);
```

- [ ] **B.** Insert on Line 04:
```
public Fish( ) {
    numFins = 3;
    numGills = 2;
}
```

- [ ] **C.** Insert on Line 21:
```
super( );
```

- [ ] **D.** Insert on Line 21:
```
super(2, 3);
```

Assistance

Show explanation | Show answer

If the super keyword is not used in a constructor of a child class, the compiler will implicitly call the no-argument constructor of the parent class using the super keyword. If a no-argument constructor is not declared in the parent class, a compiler error will occur. In this case, the no-argument constructor for the AngelFish child class does not include the super keyword in its method body. The compiler will attempt to invoke the default no-argument constructor of the Fish parent class. Since a constructor is defined in the Fish class and a no-argument constructor is not defined in the Fish class, this will cause a compiler error. To allow this code to compile, you can create the no-argument constructor in the Fish parent class that the compiler is attempting to access with the super keyword.

You can also insert the superkeyword on Line 21 to access the constructor that takes two integers as parameters in the parent Fish class. As long as the superkeyword is used in the constructor, the compiler will not use the super keyword to implicitly call a no-argument constructor that does not exist.

Question 73 of 80

Given the code:

```
01. public class MathTest {
02.     public static void main(String[ ] args) {
03.         boolean b1 = false;
04.         int x = 7;
05.         int y = 5;
06.         int z = 0;
07.         if (b1 = true) {
08.             z = x - y;
09.             System.out.println("z=" + z);
10.         }
11.         z = x + y;
12.         System.out.println("z=" + z);
13.     }
14. }
```

What is the result?

Answers

- A. z=2
- B. Line 07 causes a compilation error.
- C. Line 07 causes a runtime error.
- D. z=2
  z=12
- E. z=12

Assistance

Show explanation | Show answer

When you execute this program, Line 03 assigns the value of false to the boolean variable b1. However, when Line 07 executes, the conditional expression of the if statement assigns the value of true to b1. Therefore, the if statement evaluates to true and Lines 08 and 09 will cause z=2 to be displayed to the console window. Since this is a simple if statement, execution will continue on the line of code following the if block. This will result in Lines 11 and 12 displaying z=12 to the console window.

Question 74 of 80

Examine the following code:

```
public class Test {
   public static void main(String[ ] args) { }
}
class MyClass {
   MyClass(String myString) { }
}
class TestClass {
}
```

Which group of statements correctly identifies the constructor used by each class?

Answers

- **A.** Test uses a default constructor.
  MyClass uses a user-defined constructor.
  TestClass uses a default constructor.
- **B.** Test uses a user-defined constructor.
  MyClass uses a user-defined constructor.
  TestClass uses a user-defined constructor.
- **C.** Test uses a user-defined constructor.
  MyClass uses a user-defined constructor.
  TestClass uses a default constructor.
- **D.** Test uses a user-defined constructor.
  MyClass uses a default constructor.
  TestClass uses a user-defined constructor.

Assistance

Show explanation | Show answer

All classes must have a constructor to create an object when the program runs. A no-argument default constructor is created automatically by the Java compiler when no constructor is defined for a class.

In this case, neither the Test nor the TestClass classes have a constructor defined. Therefore the Java compiler will provide a default constructor for each of these classes. The MyClass class has a constructor defined; therefore, it will use that constructor.

Question 75 of 80

Given the following code:

```
01. String a = "Test";
02. String b = "Test";
03. // Insert code here
04.     System.out.println("refs are equal"); }
05. else{
06.     System.out.println("refs not equal"); }
```

Which two code segments, inserted independently at Line 03, will test whether the two references are referring to the same object?

Answers

- A. `if (a != b) {`
- B. `if (a.equals(b)) {`
- C. `if (a == b) {`
- D. `if (a.contains(b)) {`

Assistance

Show explanation | Show answer

The == operator is used to compare references between two objects. It will return true when the two references are referring to the same location in memory. The if (a == b) { code segment will test whether the two references point to the same object and will result in refs are equal being displayed to the console window.

The != operator is used to compare the references of two objects. It will return true if the two references are referring to objects that are not stored in the same location in memory. Therefore, it can be used to test whether the references point to the same object. The if (a != b) { code segment will result in refs not equal being displayed to the console window.

Question 76 of 80

Which object-oriented principle involves creating a class that inherits from another class and requires that the implementing class implements some of its own features?

Answers

- A. Inheritance
- B. Encapsulation
- C. Abstraction
- D. Polymorphism

Assistance

Show explanation | Show answer

Polymorphism is an object-oriented principle that refers to when a one class inherits from another class and the subclass has some of the functionality of the parent class. To create polymorphic code the subclass must implement some of its own features in addition to its inherited features.

Question 77 of 80

Given the code:

```
01. public class Looping {
02.     public static void main(String[ ] args) {
03.         int x = 0;
04.         boolean bool = true;
05.         do {
06.             while (x < 3) {
07.                 System.out.print(x + " ");
08.                 x++;
09.             }
10.             bool = false;
11.             x++;
12.         } while (bool == true);
13.         System.out.print(x);
14.     }
15. }
```

What is the result?

Answers

- A. A runtime error
- B. 0 1 2 3
- C. A compilation error
- D. 0 1 2 4
- E. 1 2 3 4

Assistance

Show explanation | Show answer

It is possible to nest loops inside other loops. In this code, the outer loop is a do-while loop. This means that the code in the body of the do-while loop on Line 05 will execute at least once. When the while statement on Line 06 is reached, the value of x is initially 0. Therefore, the body of the while loop will execute and print 0. The x variable is then incremented and the while loop executes again. Before the while loop exits, 0 1 2 will have been printed to the console window. After leaving the inner while loop, control is returned back inside the do-while loop to Line 10, where the value of bool is changed.

Also, the value of x (which is currently 3) is incremented to 4 at Line 11. When the while condition of the outer do-while loop at Line 12 is evaluated, it returns a value of false. This causes control to exit from the outer do-while loop to Line 13 and print the current value of x (which is 4) to the console window. The end result of 0 1 2 4is displayed.

Question 78 of 80

Given the following code:

```
01. ArrayList myAList = new ArrayList( );
02. myAList.add("hello");
03. myAList.add("world");
04. myAList.add(20);
05. myAList.add("20");
06. myAList.add("world");
07. System.out.println(myAList.indexOf("world"));
08. System.out.println(myAList.indexOf("Java"));
```

What is printed to the console window?

Answers

- **A.** Compilation will fail due to an error on Line 08.
- **B.** 2
  -1
- **C.** 1
  -1
- **D.** 4
  -1

Assistance

Show explanation | Show answer

The indexOf method of the ArrayList class takes an object as a parameter and returns the index of the first occurrence of the object it is passed. In this case, the indexOf method on Line 07 will return the index of the first occurrence of world in myAList. The first occurrence of world is located in the second position of myAList. Since an ArrayList is zero-index based, 1 will be printed to the console window.

Then, the indexOf method on Line 08 tries to find the first occurrence of Java. However, Java does not exist in myAList. When the indexOfmethod cannot find an occurrence of the object it is passed, it returns -1. Therefore, -1 will be printed to the console window.

Question 79 of 80

You are creating a calendar program that allows users to enter a date. The date will be entered as a string in the "MMM d yyyy" format and will be stored in a String variable named input. You need to convert this input to a LocalDate object so it can be used with the program. Which code segment should you use?

Answers

A.
```
DateTimeFormatter formatter =
    DateTimeFormatter.ofPattern(
    "MMM d yyyy");
LocalDate date = LocalDate.
    parse(input, formatter);
```

B.
```
LocalDate date = new LocalDate(input);
```

C.
```
LocalDate date = LocalDate.now(input);
```

D.
```
DateTimeFormatter formatter =
    new DateTimeFormatter("MMM d yyyy");
LocalDate date = LocalDate.
    parse(input, formatter);
```

Assistance

Show explanation | Show answer

The DateTimeFormatter class can be used to format a date. However, this class does not have any public constructors. An instance of DateTimeFormatter can be created with the ofPattern method. The format that you want to use can be specified as a string in this method.

The parse method of the LocalDate class can be used to convert a string value to a LocalDate object using a specified DateTimeFormatter object. The version of the parse method that takes two arguments must be passed the date as a string and the DateTimeFormatter object, in that order.

Question 80 of 80

You are creating a Java program that will allow users to manage important dates on a calendar. You have created the following code:

```
01. DateTimeFormatter formatter =
02.     DateTimeFormatter.ofPattern("d M yyyy");
03. LocalDate date = LocalDate.of(2015, 12, 10);
04. System.out.println(date.format(formatter));
```

What is the result?

Answers

- **A.** 42289
- **B.** 2015 10 12
- **C.** 42289
- **D.** 10 12 2015

Assistance

Show explanation | Show answer

The ofPattern method of the DateTimeFormatter class can be used to generate a DateTimeFormatter that uses a specific pattern. The d, M, and y symbols are reserved pattern letters that can be used to define a pattern. The d symbol represents the day of month as a number, the M symbol represents the month of year as text or a number, and the y symbol represents the year of era as a year. The number of times a symbol is used in a pattern will determine how a date will be formatted. The "d M yyyy" pattern will return a single number for the day of month, a single number for the month of year, and the year of era using four values.

The format method of the LocalDate class can be used to obtain a formatted version of the date as a String object. This method must be passed a valid DateTimeFormatter to specify how the date should be formatted. In this case, the LocalDate object on Line 03 is created with a year of 2015, a month of 12, and a day of 10, in that order. Invoking the format method through this LocalDate object on Line 04 and passing it a DateTimeFormatter object that uses the "d M yyyy" pattern will return 10 12 2015.

Question 1 of 80

Given the following code:

```
try {
  ...
}
catch(IOException e) {
  ...
}
```

Which two advantages of exceptions are being used?

Answers

- [ ] **A.** Removing errors from the program
- [ ] **B.** Separating the code that handles errors from other code
- [x] **C.** Grouping and differentiating error types for easier handling
- [ ] **D.** Preventing errors from being propagated up the call stack

Assistance

Show explanation | Show answer

A try-catch statement allows you to separate logic code from code that is designed to handle errors that may occur. You place any code that may cause an error in the try block. If an error occurs, execution will jump to the catch block. You must specify the type of exception that the catch block will handle. This will allow you to catch exceptions of that type, or any exception that is a subclass of that type.

In this case, the program catches any exceptions of type IOException, as well as any exceptions that are a subclass of the IOExceptionclass.

Question 2 of 80

Given the code:

```
01. public class Program {
02.     public static void main (String[] args) {
03.         int num = 16;
04.
05.         System.out.println(myMethod(num));
06.
07.     }
08.
09.     public static int myMethod(Integer num) {
10.         return (int)Math.sqrt(num);
11.     }
12. }
```

What is the result?

Answers

- A. A runtime error will occur due to the code on Line 05.
- B. A compiler error will occur due to the code on Line 05.
- C. 4
- D. 2

Assistance

Show explanation | Show answer

When a primitive type is passed into an object, the Java compiler will create an object of an appropriate type and store the value in the new object. The primitive int value stored in the num variable is passed to myMethod on Line 05. Since this method expects an Integervalue, the Java compiler automatically creates an object of type Integer and stores the value of num in the new object.

The sqrt method of the Math class returns a rounded square root of a double value. The code on Line 10 will execute and return the square root of 16, which is 4.0. Since the return value is casted as an int, the decimal will be truncated and 4 will be returned and printed to the console window.

Question 3 of 80

You want to use the class Tester in your program. Tester is located in a package named com.utilities.myclasses. You do not want your program to be able to access any other classes from the com.utilities.myclasses package. Which statement should you use?

Answers

- A. `import com.utilities.myclasses.*;`
- B. `import Tester;`
- C. `import com.utilities.myclasses.Tester;` (selected)
- D. `import com.utilities.myclasses.Tester.java;`

Assistance

Show explanation | Show answer

The Java import statement allows you to use a package or a class from another package in a class that you are writing. When using the import statement, you specify the package name and the class name to which you want to access. If you need to import more than one class, your program can have multiple import statements.

Question 4 of 80

Which two code segments correctly declare an ArrayList named myArrayList?

Answers

- [ ] A. `ArrayList myArrayList = new ArrayList;`
- [ ] B. `ArrayList myArrayList = new ArrayList( );`
- [ ] C.
```
String ArrayList myArrayList =
    new ArrayList( );
```
- [ ] D. `ArrayList myArrayList = new ArrayList(15);`

Assistance

Show explanation | Show answer

An ArrayList has an initial capacity of ten elements of type Object when it is created using the no-argument constructor. If you are not sure how big of an ArrayList you need to create, keep in mind that the ArrayList will resize itself as more elements are added. You can use a constructor that takes an integer value as a parameter to indicate the initial size of the ArrayList.

Question 5 of 80

Given the code:

```
01. public class Tester {
02.     public static void main(String[ ] args) {
03.         int x = 0;
04.         int i = 0;
05.         for ( ; i < 5; i++) {
06.             x++;
07.             if (x == 3) {
08.                 break;
09.             }
10.         }
11.         System.out.println("x is: " + x);
12.         System.out.println("i is: " + i);
13.     }
14. }
```

What is the result?

Answers

- A. x is: 3
  i is: 3
- B. Compilation error at Line 05
- C. x is: 3
  i is: 2
- D. x is: 4
  i is: 3

Assistance

Show explanation | Show answer

The condition in the if statement on Line 07 will evaluate to true when the value of x is 3. When that occurs, the break statement on Line 08 will execute. The break statement in Java causes execution to break out of an enclosing while, do-while, for, or switch statement. In this case, when x is 3, the value of i is 2. The break statement causes execution to jump out of the for loop and continue at Line 11, which displays the results to the console window.

Question 6 of 80

Which two statements describe how to include encapsulation when designing a class for your Java programs?

Answers

- [ ] **A.** Create public accessor and mutator methods for all member fields.
- [ ] **B.** Create private accessor and mutator methods for all member fields.
- [ ] **C.** Declare all member fields with the private access level modifier.
- [ ] **D.** Declare all member fields with the public access level modifier.

Assistance

Show explanation | Show answer

Encapsulation refers to the practice of controlling access to, and guarding the state of, objects in your program. One step in providing encapsulation for your classes is to declare all member fields as private members. This ensures that they can be accessed only from within the class itself.

In addition to declaring all member fields as private, a tightly encapsulated class should make use of public accessor and mutator methods. This allows you to control how the values of the private fields are accessed and manipulated. The private member fields should only be accessible via the publicly available methods.

Question 7 of 80

Given the following array declaration:

```
Object[ ][ ] myArray;
```

Which two code segments, taken independently, instantiate a multi-dimensional array containing four arrays of length 10?

Answers

- [ ] **A.** `myArray = new String[10][4];`
- [ ] **B.** `myArray = new Object[4][10];`
- [ ] **C.** `myArray = new String[4][10];`
- [ ] **D.** `myArray = new Object[10][4];`

Assistance

Show explanation | Show answer

The contents of the first set of brackets in a multi-dimensional array indicate how many arrays will be created. The sets of brackets after the first set indicate the dimensions of the array.

In this case, there are only two sets of square brackets. The first set indicates how many arrays will be stored in myArray while the second set indicates the length of the arrays. The new keyword can be used to instantiate an array of a specified data type. The following code segment correctly instantiates an array of type Object that contains four arrays with a length of 10.

```
myArray = new Object[4][10];
```

The myArray multi-dimensional array is declared as an Object and instantiated as a String. This is possible because the String class inherits from the Object class. However, this will make it so myArray can only store String objects rather than Object objects.

Question 8 of 80

You are creating a Java program that will display whether a student passed or failed their exam. To pass the exam, a student must score at least 75 percent. Your program includes the following code for testing purposes:

```
01. int score = 78;
02. // Insert code here
03.     System.out.print("You Passed!");
04. } else {
05.     System.out.print("You did not pass.");
06. }
```

Which code segment should you insert on Line 02?

Answers

- A. `if (score >= 75) {`
- B. `if (score < 74) {`
- C. `if (score <= 74) {`
- D. `if (score > 75) {`

Assistance

Show explanation | Show answer

One of the most basic of conditional branch statements is the if-else statement. When the boolean expression of the if statement evaluates to true, the if block of statements is executed. When the boolean expression of the if statement evaluates to false, the elseblock of statements is executed. In this case, a student requires a score of 75 percent or more to pass the test.

Therefore, the conditional expression must check the score variable for a value of 75 or more (>=) for the if block to display that the student passed.

Question 9 of 80

Given the code:

```
01.  public class Employee {
02.      private String empName;
03.      private int empNum;
04.      private String empDept;
05.
06.      public Employee( ) {
07.          // Insert code here
08.      }
09.
10.      public Employee(String name, int num, String dept) {
11.          empName = name;
12.          empNum = num;
13.          empDept = dept;
14.      }
15.
15.  ...
16.
17.  }
```

When the no-argument constructor is called, you want to assign the values of "Unknown", 55, and "Unknown" to the empName, empNum, and empDept fields, respectively. Which line of code should you insert on Line 07?

Answers

- A. `Employee("Unknown", 55, "Unknown");`
- B. `this("Unknown", 55, "Unknown");`
- C. `super("Unknown", 55, "Unknown");`
- D. `base("Unknown", 55, "Unknown");`

Assistance

Show explanation | Show answer

You can call one constructor from within a different constructor in the same class using the this keyword. Such a call is known as an explicit constructor invocation. If you have a constructor that uses an explicit constructor invocation, it must be the first line of code in the constructor. The compiler will determine which constructor is being called based on the number and type of the arguments.

The this("Unknown", 55, "Unknown"); line of code will call the constructor on Line 10, passing it the three arguments. Execution of the three argument constructor will result in the values passed as arguments being assigned to the three fields of the Employee class.

Question 10 of 80

Given the code:

```
01. public class CountInput {
02.     public static void main(String[ ] args) {
03.         int total = args.length;
04.         switch (total) {
05.             case 0:
06.             case 1:
07.             case 2:  System.out.println("2 or less");
08.                      break;
09.             case 3:
10.             case 4:
11.             case 5:  System.out.println("5 or less");
12.                      continue;
13.             default: System.out.println("More than 5");
14.         }
15.         System.out.println("Finished");
16.     }
17. }
```

What is the result?

Answers

- A. 5 or less
  More than 5
  Finished
- B. Compilation fails due to an error on Line 05.
- C. More than 5
  Finished
- D. Line 05 causes a runtime error.
- E. Compilation fails due to an error on Line 12.

Assistance

Show explanation | Show answer

Attempting to compile this program will cause a continue outside of loop error on Line 12. The continue statement is used to skip the remainder of the current iteration of a loop. The continue statement can be used in for, while, and do-while loops. However, you cannot use the continue statement with a switch statement.

Question 11 of 80

Given the code:

```
01. class Person {
02. private int age;
03. private String name;
04.
05.     public String getName( ) {
06.         return name;
07.     }
08.     public void setName(String perName) {
09.         name = perName;
10.     }
11. }
```

What should you do to ensure that the age variable adheres to the principles of tight encapsulation?

Answers

- **A.** Change the modifier on all methods in the Person class to private, and insert this code segment on Line 04:

```
private int getAge( ) { return age;  }
private void setAge(int perAge) {
  age = perAge;
}
```

- **B.** Insert this code segment on Line 04:

```
public int getAge( ) {
  return age;
}
public void setAge(int perAge) {
  age = perAge;
}
```

- **C.** Change the modifier on the age variable to public.
- **D.** Change the modifier on the getName and setName methods to private.

Assistance

Show explanation | Show answer

For a class to follow the principles of tight encapsulation, direct access to the fields must be restricted to the class in which they are declared. Any changes to the fields must be done through public methods that belong to the same class as the fields.

Since the age variable is declared as private, it cannot be modified outside of the Person class. To resolve this and ensure tight encapsulation, you must create a public method that allows the age variable to be modified. In addition, you must create a public method that can retrieve the current value stored in the age variable.

Question 12 of 80

Which keyword can appear only once in a single source file?

Answers

- A. import
- B. package
- C. final
- D. static
- E. class

Assistance

Show explanation | Show answer

You use the package keyword to create a package that includes your class or classes. The package keyword is followed by the name of the package and it must be the first line of code in the source file. A Java source file can contain, at most, a single package statement. If no package statement is present, then the class is said to belong to the default package.

Question 13 of 80

You need to create several primitive variables at the class level. Which two names are valid when declaring a variable?

Answers

- [ ] **A.** `int value+;`
- [ ] **B.** `double 2x;`
- [ ] **C.** `String _name;`
- [ ] **D.** `double x2;`

Assistance

Show explanation | Show answer

Java naming conventions specify that a variable name can contain the underscore character. In fact, the variable name can start with a letter, an underscore, or a dollar sign. Using _name as a variable name is valid and will compile and run successfully. Once a variable name starts with a valid character, you can use letters, numbers, dollar signs, or the underscore in subsequent characters. Using x2 as a variable name is valid and will compile and run successfully.

Question 14 of 80

In which scenario would it be best to use a switch statement?

Answers

- A. A Yes or No response by the user determines whether the program terminates.
- B. Users are sorted based on their letter grade.
- C. A Boolean value determines which block of code executes.
- D. A block of code executes only when a value of 150 or more is entered as the user's weight.
- E. The gender specified by the user determines which survey they will receive.

Assistance

Show explanation | Show answer

A switch statement is a branching statement that can have a number of possible paths of execution. The value of the switch statement's expression determines which block of code in the switch statement executes. In this case, there are five possible letter grades that can be assigned to a user. Since you only need to sort users based on their letter grade, you only need to execute one block of code for each user to sort them appropriately. A switch statement is an alternative to using a series of if-else statements.

Question 15 of 80

Which two code segments, taken independently, correctly declare a multi-dimensional array of type int named anArray?

Answers

- A. `int[ ][ ] anArray;`
- B. `int[ ][ ][ ] anArray;`
- C. `int anArray;`
- D. `int[ ] anArray;`

Assistance

Show explanation | Show answer

A multi-dimensional array is an array that contains other arrays. The maximum number of dimensions for a multi-dimensional array is 255. The contents of the multi-dimensional array must match the declared data type.

When three sets of brackets are used to declare an array, the first set of square brackets indicates how many two-dimensional arrays are contained in the multi-dimensional array. The second set of brackets indicates how many one-dimensional arrays are stored in each of the two-dimensional arrays. The last set of brackets indicates the length of each array.

When two sets of brackets are used to declare an array, the first dimension indicates how many one-dimensional arrays will be stored in the multi-dimensional array. The second dimension indicates the length of each array.

Question 16 of 80

Given the code:

```
01. public class Test {
02.
03.     interface Calculations {
04.         int calculation(int a, int b);
05.     }
06.
07.     public static int performCalculation(int a, int b, Calculations calc) {
08.         return calc.calculation(a, b);
09.     }
10.
11.     public static void main (String[ ] args ) {
12.         Calculations multiplication = (int a, int b) -> a * b;
13.         Calculations subtraction = (int a, int b) -> a - b;
14.         System.out.println(performCalculation(5, 3, multiplication) + "\n" + performCalculation(5, 3, subtracti
15.     }
16. }
```

What is the result?

Answers

- A. A runtime error will occur due to the code on Line 14.
- B. 15 2
- C. A compiler error will occur due to the code on Line 14.
- D. 15
  2

Assistance

Show explanation | Show answer

The multiplication and subtraction Calculations objects are each assigned a lambda expression that uses a parameter list matching the parameter list of the calculation method of the Calculations interface. The lambda expression assigned to multiplicationmultiplies two int values, while the lambda expression assigned to subtraction subtracts two int values. The code on Line 14 executes the performCalculation method twice.

Since the first execution of performCalculation is passed 5, 3, and the multiplication object, 15 will be printed to the console window. The second execution of performCalculation is executed after the newline character and is passed 5, 3, and the subtraction object. Therefore, 2 will be printed on a new line in the console window.

Question 17 of 80

Given the code:

```
01. import java.io.*;
02. import java.text.*;
03.
04. class Parent {
05.      // Insert code here
06.     {
07.          ...
08.     }
09. }
10.
11. class Child extends Parent {
12.     public void doSomething( ) throws ParseException
13.     {
14.         ...
15.     }
16. }
```

Which two method declarations, when inserted independently on Line 05, will allow the code to compile without error?

Answers

- A. `public void doSomething( ) throws Exception`
- B. `public void doSomething( ) throws RuntimeException`
- C. `public void doSomething( )`
- D. `public void doSomething( ) throws IOException`
- E. `public void doSomething( ) throws ParseException`

Assistance

Show explanation | Show answer

When an overriding method throws an exception, the exception must be the same as, or a subclass of, the exception thrown by the method in the super class. In this case, the method on Line 12 throws a ParseException. Declaring a doSomething method in the super class that throws the same ParseException is valid Java code and will compile successfully.