QUIZ

What would the following code display?

```
int[] a = {1,2,3,4,5,6};
int i = a.length - 1;
while(i>=0){
    System.out.print(a[i]);
    i--;
}
```

a. 123456

b. An exception could be thrown at runtime.

c. 654321

d. nothing

e. 65432

f. 12345

What would the following code display?

```
int[] a = {1,2,3,4,5,6};
int i = a.length - 1;
while(i>=0){
    System.out.print(a[i]);
    i--;
}
```

a. 123456

b. An exception could be thrown at runtime.

c. 654321

d. nothing

e. 65432

f. 12345

A and E are classes, B and D are interfaces
Which would be correct?

a. `interface F implements B,D { }`
b. `interface F implements D { }`
c. `interface F extends D { }`
d. `interface F extends E { }`
e. `interface F extends B,D { }`

A and E are classes, B and D are interfaces
Which would be correct?

a. `interface F implements B,D { }`
b. `interface F implements D { }`
c. `interface F extends D { }`
d. `interface F extends E { }`
e. `interface F extends B,D { }`

What is inheritance?

a. It is the process where one object acquires the properties of another.
b. inheritance is the ability of an object to take on many forms.
c. inheritance is a technique to define different methods of same type.
d. None of the above.

What is inheritance?

a. It is the process where one object acquires the properties of another.
b. inheritance is the ability of an object to take on many forms.
c. inheritance is a technique to define different methods of same type.
d. None of the above.

What is polymorphism?

a. Polymorphism is a technique to define different objects of same type.
b. Polymorphism is the ability of an object to take on many forms.
c. Polymorphism is a technique to define different methods of same type.
d. None of the above

What is polymorphism?

a. Polymorphism is a technique to define different objects of same type.

b. Polymorphism is the ability of an object to take on many forms.

c. Polymorphism is a technique to define different methods of same type.

d. None of the above

What are instance variables?

a. Instance variables are static variables within a class but outside any method.
b. Instance variables are variables defined inside methods, constructors or blocks.
c. Instance variables are variables within a class but outside any method.
d. None of the above.

What are instance variables?

a. Instance variables are static variables within a class but outside any method.
b. Instance variables are variables defined inside methods, constructors or blocks.
c. Instance variables are variables within a class but outside any method.
d. None of the above.

When people refer to "a member of a class", they mean:

a. An attribute
b. A method
c. Attribute or method
d. A sub-class

When people refer to "a member of a class", they mean:

a. An attribute
b. A method
c. Attribute or method
d. A sub-class

What would display the following statements?

```
int[] nums = {1,2,3,4,5,6};
System.out.println((nums[1] + nums[3]));
```

a. 6

b. 2+4

c. 1+3

d. 4

What would display the following statements?

```
int[] nums = {1,2,3,4,5,6};
System.out.println((nums[1] + nums[3]));
```

a. 6

b. 2+4

c. 1+3

d. 4

If classes Student, Staff and Faculty extend class Person, which one makes sense:

a. Faculty[] faculties={new Person(),
new Staff(), new Student()};
b. Staff[] staff={new Person(),
new Faculty(), new Student()};
c. Person[] persons={new Faculty(),
new Staff(), new Student()};

If classes Student, Staff and Faculty extend class Person, which one makes sense:

a. Faculty[] faculties={new Person(),
new Staff(), new Student()};
b. Staff[] staff={new Person(),
new Faculty(), new Student()};
c. Person[] persons={new Faculty(),
new Staff(), new Student()};

What is the output of the following program?

```
class Recursion {

  int func(int n) {
    int result;
     if (n == 0) {
        result = 3;
     } else {
        result = func(n - 1);
     }
     return result;
  }
}

public class Prog {
  public static void main(String args[]) {
     Recursion obj = new Recursion() ;
     System.out.print(obj.func(5));
  }
}
```

a. 5
b. 18
c. 3
d. Compilation Error
e. Runtime Error

What is the output of the following program?

```
class Recursion {

  int func(int n) {
    int result;
     if (n == 0) {
        result = 3;
     } else {
        result = func(n - 1);
     }
     return result;
  }
}

public class Prog {
  public static void main(String args[]) {
     Recursion obj = new Recursion() ;
     System.out.print(obj.func(5));
  }
}
```

a. 5
b. 18
c. 3
d. Compilation Error
e. Runtime Error

What is the output of the following program?

```
class Recursion {
   int func(int n) {
      int result = 0;
      if (n < 10) {
        if (n == 8) {
          result = 1;
        }
      } else {
        result = func(n % 10) + func(n / 10);
      }
      return result;
   }
}
public class Prog {
   public static void main(String args[]) {
       Recursion obj = new Recursion() ;
       System.out.print(obj.func(123) + " ");
       System.out.print(obj.func(8328) + " ");
       System.out.println(obj.func(8325));
   }
}
```

a. 0 0 0
b. 0 2 1
c. 0 1 0
d. Compilation Error

What is the output of the following program?

```
class Recursion {
   int func(int n) {
      int result = 0;
      if (n < 10) {
        if (n == 8) {
          result = 1;
        }
      } else {
        result = func(n % 10) + func(n / 10);
      }
      return result;
   }
}
public class Prog {
   public static void main(String args[]) {
       Recursion obj = new Recursion() ;
       System.out.print(obj.func(123) + " ");
       System.out.print(obj.func(8328) + " ");
       System.out.println(obj.func(8325));
   }
}
```

a. 0 0 0

b. 0 2 1

c. 0 1 0

d. Compilation Error

Which of the following is a valid annotation definition ?

a. public @annotation MyAnnotation{ }
b. private @interface MyAnnotation{ }
c. public @interface MyAnnotation{ }
d. public @MyAnnotation{ }

Which of the following is a valid annotation definition ?

a. public @annotation MyAnnotation{ }
b. private @interface MyAnnotation{ }
c. public @interface MyAnnotation{ }
d. public @MyAnnotation{ }

Which of the following belongs to meta-annotations?

a. @Override
b. @Retention
c. @Deprecated
d. @SuppressWarnings()

Which of the following belongs to meta-annotations?

a. @Override
b. @Retention
c. @Deprecated
d. @SuppressWarnings()

Which of the following are valid retention policy types in Java (multiple answers)?

a. SOURCE
b. CLASS
c. RUNTIME
d. CODE
e. TOOLS

Which of the following are valid retention policy types in Java (multiple answers)?

a. SOURCE
b. CLASS
c. RUNTIME
d. CODE
e. TOOLS

What will be the output of the following program?

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.RUNTIME)
@interface MyAnnotation {
  int value();
}

class Hello {
  @MyAnnotation(value = 10)
  public void goodEvening() {
    System.out.print("Good Evening");
  }
}
public class HelloAnnotation {
 public static void main(String args[]) throws Exception {
   Hello h = new Hello();
   Method m = h.getClass().getMethod("goodEvening");
   MyAnnotation myAnn = m.getAnnotation(MyAnnotation.class);
   System.out.println("Prints " + myAnn.value());
 }
}
```

a. Prints HelloAnnotation 10
b. Good Evening Prints 10
c. Prints 10
d. Some other output
e. Output cannot be determined

What will be the output of the following program?

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.RUNTIME)
@interface MyAnnotation {
  int value();
}

class Hello {
  @MyAnnotation(value = 10)
  public void goodEvening() {
    System.out.print("Good Evening");
  }
}
public class HelloAnnotation {
 public static void main(String args[]) throws Exception {
   Hello h = new Hello();
   Method m = h.getClass().getMethod("goodEvening");
   MyAnnotation myAnn = m.getAnnotation(MyAnnotation.class);
   System.out.println("Prints " + myAnn.value());
 }
}
```

a. Prints HelloAnnotation 10
b. Good Evening Prints 10
c. Prints 10
d. Some other output
e. Output cannot be determined

What will be the output of the following program?

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.CLASS)
@interface MyAnnotation {
   int value();
}
class Hello {
  @MyAnnotation(value = 10)
  public void goodEvening() {
    System.out.print("Good Evening");
  }
}
public class HelloAnnotation {
  public static void main(String args[]) throws Exception {
    Hello h = new Hello();
    Method m = h.getClass().getMethod("goodEvening");
    MyAnnotation myAnn = m.getAnnotation(MyAnnotation.class);
    System.out.println("Prints " + myAnn.value());
  }
}
```

a. Prints HelloAnnotation 10
b. Good Evening Prints 10
c. Prints 10
d. Some other output
e. Output cannot be determined
f. Compilation Error
g. Runtime Exception

What will be the output of the following program?

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.CLASS)
@interface MyAnnotation {
   int value();
}
class Hello {
  @MyAnnotation(value = 10)
  public void goodEvening() {
    System.out.print("Good Evening");
  }
}
public class HelloAnnotation {
  public static void main(String args[]) throws Exception {
    Hello h = new Hello();
    Method m = h.getClass().getMethod("goodEvening");
    MyAnnotation myAnn = m.getAnnotation(MyAnnotation.class);
    System.out.println("Prints " + myAnn.value());
  }
}
```

a. Prints HelloAnnotation 10
b. Good Evening Prints 10
c. Prints 10
d. Some other output
e. Output cannot be determined
f. Compilation Error
g. Runtime Exception

What is the output of the following program?

```
interface Int{}
class Simple implements Int{}
class SimpleTest {
   public static void main(String[] args) {
     try {
       Class c=Class.forName("Simple");
       System.out.print(c.isInterface() + " ");
       c=Class.forName("Int");
       System.out.println(c.isInterface());
     } catch (Exception e) {
       System.out.println(e);
     }
   }
}
```

a. false true
b. true true
c. true false
d. false false

What is the output of the following program?

```
interface Int{}
class Simple implements Int{}
class SimpleTest {
   public static void main(String[] args) {
     try {
       Class c=Class.forName("Simple");
       System.out.print(c.isInterface() + " ");
       c=Class.forName("Int");
       System.out.println(c.isInterface());
     } catch (Exception e) {
       System.out.println(e);
     }
   }
}
```

a. false true
b. true true
c. true false
d. false false

Which of the following can obtain the location of file Reflection.class at runtime?

public class Reflection {}

a. Reflection.class.getClassLoader().getResource("Reflection.class")
b. Reflection.getClass().getResource("Reflection.class")
c. Reflection.getClass().getClassLoader().getResource("Reflection.class")
d. Reflection.class.getClassLoader("Reflection.class")

Which of the following can obtain the location of file Reflection.class at runtime?

public class Reflection {}

a. Reflection.class.getClassLoader().getResource("Reflection.class")
b. Reflection.getClass().getResource("Reflection.class")
c. Reflection.getClass().getClassLoader().getResource("Reflection.class")
d. Reflection.class.getClassLoader("Reflection.class")

Consider the following program:

```
class MyClass {
   private float val;
   @Deprecated
   MyClass(int n) {
     val = n;
   }
   MyClass(float f) {
     val = f;
   }
}

public class TestDeprecated {
   public static void main(String[] args) {
     int val = 3;
     MyClass obj = new MyClass(3);
   }
}
```

What of the following is true?

a. It generates one warning (call of the deprecated constructor) when compiling, nothing when running
b. It generates two warnings (definition of the deprecated constructor and call of the deprecated constructor) when compiling, nothing when running
c. No warning when compiling but it generates one warning when running
d. No warning when compiling but it generates two warnings when running
e. No warning when compiling, exception when running
f. One warning when compiling, exception when running

What of the following is true?

a. It generates one warning (call of the deprecated constructor) when compiling, nothing when running
b. It generates two warnings (definition of the deprecated constructor and call of the deprecated constructor) when compiling, nothing when running
c. No warning when compiling but it generates one warning when running
d. No warning when compiling but it generates two warnings when running
e. No warning when compiling, exception when running
f. One warning when compiling, exception when running

What is the main function of JDBC?

a. Create a connection to the database.
b. Send SQL statements to the database.
c. Processing data and query results
d. All of the above

What is the main function of JDBC?

a. Create a connection to the database.
b. Send SQL statements to the database.
c. Processing data and query results
d. All of the above

(multiple choice) which of the following steps are performed when accessing the database through JDBC?

a. Loading the JDBC driver.
b. Setting up the connection.
c. Executing queries or applying changes to the database
d. Close the connection

(multiple choice) which of the following steps are performed when accessing the database through JDBC?

a. Loading the JDBC driver.
b. Setting up the connection.
c. Executing queries or applying changes to the database
d. Close the connection

Which package allows Java to access a database?

a. java.sql
b. java.db
c. java.dbms
d. java.jdbc
e. java.lang
f. java.util

Which package allows Java to access a database?

a. java.sql
b. java.db
c. java.dbms
d. java.jdbc
e. java.lang
f. java.util

In general, you prefer adding a TableView to a:

a. BorderPane
b. StackPane
c. ScrollPane
d. SplitPane
e. TabPane
f. TitledPane

In general, you prefer adding a TableView to a:

a. BorderPane
b. StackPane
c. ScrollPane
d. SplitPane
e. TabPane
f. TitledPane

FXCollections:

a. Is a package that contains classes for collections that you should use as backend to some JavaFX widgets.
b. Is a class that contains wrapper methods for regular collections so that you can use them as backend to some JavaFX widgets.
c. Is a class that only contains the ObservableList inner class
d. Is an enum that is used for defining the collections that can be used with JavaFx widgets (for instance a TreeSet cannot be used, and doesn't appear in the enum)

FXCollections:

a. Is a package that contains classes for collections that you should use as backend to some JavaFX widgets.

b. Is a class that contains wrapper methods for regular collections so that you can use them as backend to some JavaFX widgets.

c. Is a class that only contains the ObservableList inner class

d. Is an enum that is used for defining the collections that can be used with JavaFx widgets (for instance a TreeSet cannot be used, and doesn't appear in the enum)

"static" in java indicates:

a. Something that you cannot change
b. Something that is attached to the class, not to a particular object instance
c. Something that cannot throw exceptions
d. Something that cannot be overriden (methods) or extended

"static" in java indicates:

a. Something that you cannot change
b. Something that is attached to the class, not to a particular object instance
c. Something that cannot throw exceptions
d. Something that cannot be overriden (methods) or extended

The protocol that takes a message from a network to another network is:

a. FTP
b. TCP
c. IP
d. HTTP

The protocol that takes a message from a network to another network is:

a. FTP
b. TCP
c. IP
d. HTTP

Which tool isn't a build tool:

a. ant
b. junit
c. make
d. maven

Which tool isn't a build tool:

a. ant
b. junit
c. make
d. maven

The time taken by a message to travel between computers is called:

a. bandwidth
b. routing
c. latency
d. delay

The time taken by a message to travel between computers is called:

a. bandwidth
b. routing
c. latency
d. delay

You need to have in your classpath the .jar of the JDBC driver that you are using:

a. Only when you compile the program
b. Only when you run the program
c. Both when you compile and run the program

You need to have in your classpath the .jar of the JDBC driver that you are using:

a. Only when you compile the program
b. Only when you run the program
c. Both when you compile and run the program

If you serialize an object that contains references, when you reload it the referenced objects will go at exactly the same place in memory:

a. True
b. False

If you serialize an object that contains references, when you reload it the referenced objects will go at exactly the same place in memory:

a. True
b. False

To make an object serializable:

a. You don't need to do anything - all objects are serializable by default.
b. You only need to say that it implements the Serializable interface.
c. You need to say that it implements the Serializable interface and implement the two methods writeObject() and readObject() defined in the interface.

To make an object serializable:

a. You don't need to do anything - all objects are serializable by default.

b. You only need to say that it implements the Serializable interface.

c. You need to say that it implements the Serializable interface and implement the two methods writeObject() and readObject() defined in the interface.