2. Classes in Java Results for Christine En-Tse Cheng

Correct answers are hidden.

Submitted Dec 13 at 5:20p.m.

Unanswered

Question 1

0 / 1 pts

What is the difference between a class variable and an instance variable in Java?

- Class variables are declared using private, while instance variables use public.
- Class variables must be declared inside methods; instance variables are declared outside.
- Class variables are shared across all instances; instance variables are unique to each instance of the class.
- There is no difference; both terms mean the same thing.

Unanswered

Question 2

0 / 1 pts

What keyword is used in Java to refer to the current object inside an instance method or constructor?

- object
- self
- instance
- this

Unanswered

Question 3

0 / 1 pts

Which of the following is **not** an access modifier in Java?

- internal
- public
- protected
- private

Unanswered

Question 4

0 / 1 pts

What is a constructor in Java? Choose the best answer.

- A method that initializes an object when it is created.
- A static method used to build a class.
- A method that returns a new object.
- A method that must return void.

Quiz Submissions

Attempt 1: 0

This quiz has unlimited attempts

← Back to Quiz

Unanswered

Question 5

0 / 1 pts

Which of the following best describes the order of operations when creating a new object in Java?

- ○ Instance variables are initialized, then the constructor runs.
- ○ Static variables are initialized after the constructor is called.
- ○ The superclass constructor runs after the subclass constructor finishes.
- ○ The constructor runs first, then instance variables are initialized

Unanswered

Question 6

0 / 1 pts

Which of the following code fragments demonstrate proper overloading (i.e. overloads a method and does not result in an error)?

☐
```
class A {
  void func(double d) {
    System.out.println(d);
  }

  void func(String s) {
    System.out.println(s);
  }
}
```

☐
```
class A {
  void func(double d, String s) {
    System.out.println(d);
  }

  void func(String s, double d) {
    System.out.println(d);
  }
}
```

☐
```
class A {
  void func(double d) {
    System.out.println(d);
  }

  void func(double d2) {
    System.out.println(d2);
  }
}
```

☐
```
class A {
  void func(double d) {
    System.out.println(d);
  }

  double func(double d) {
    return d;
  }
}
```

☐
```
class A {
  void func(double d) {
    System.out.println(d);
  }

  void func(double d1, double d2) {
    System.out.println(d1 + d2);
  }
}
```

Unanswered

Question 7

0 / 1 pts

Select the correct statement(s) below about the `toString` method.

- ☐ `toString` is a method in class `Object`, therefore, we should never modify it.
- ☐ Classes, especially subclasses, often override the `toString` method to give a concise textual representation of their instances.
- ☐ The `toString` method should return a string representation of an object.
- ☐ All Java classes inherit the `toString` method from the `Object` class.

Unanswered

Question 8

0 / 1 pts

In Java, what does the `==` operator check when its arguments are objects?

- ○ Whether the objects are equal according to the `equals` method.
- ○ Whether the objects are of the same type.
- ○ Whether the objects have the same memory location.
- ○ Whether the objects have the same values.

Unanswered

Question 9

0 / 1 pts

Suppose we have the following lines of code:

```
String string1 = new String("butterfly A");
String string2 = new String("butterfly B");
String string3 = string2;
```

Select the statements that would produce `true`.

- ☐ "butterfly A" == string1
- ☐ string3.equals(string2)
- ☐ string3 == string1
- ☐ string1 == string2
- ☐ string1.equals(string2)
- ☐ "butterfly" == "butterfly"
- ☐ string3 == string2
- ☐ string1.equals(string3)

Unanswered

Question 10

0 / 1 pts

Assume we have a `Coordinate` class with two instance variables, `x` and `y`.

Suppose we override the `hashCode` method with our own and do not override `equals`:

```
  /**
  * The hash code of this instance will be the sum of the coordinates.
  * @return x + y
  */
  @Override
  public int hashCode() {
    return x + y;
  }
```

Consider the code below:

```
public static void main(String[] args) {
  Coordinate pointOne = new Coordinate(1, 2);
  Coordinate pointTwo = new Coordinate(1, 2);
  Coordinate pointThree = new Coordinate(3, 4);
  Coordinate pointFour = new Coordinate(2, 1);
}
```

Select the correct statement(s) from below.

- ☐ `pointTwo.hashCode()` and `pointFour.hashCode()` return the same value. Since they are different objects, this means that the implementation of `hashCode` must be incorrect.
- ☐ `pointOne.hashCode()` and `pointTwo.hashCode()` return the same value.
- ☐ `pointOne.equals(pointTwo)` returns `false`.

Unanswered

Question 11

0 / 1 pts

Which of the following code fragments do not result in an error?

```
public class Adder {
    private static int a = 0;
    private int b = 0;

    public void func() {
        b++;
    }
}
```

```
public class Adder {
    private static int a = 0;
    private int b = 0;

    public static void func() {
        b++;
    }
}
```

```
public class Adder {
    private static int a = 0;
    private int b = 0;

    public void func() {
        a++;
    }
}
```

```
public class Adder {
    private static int a = 0;
    private int b = 0;

    public static void func() {
        a++;
    }
}
```

```
public class Adder {
    private static int a = 0;
    private int b = 0;

    public static void func() {
        a++;
    }

    public static void main(String[] args) {
        Adder.func();
    }
}
```

```
public class Adder {
    private static int a = 0;
    private int b = 0;

    public void func() {
        a++;
    }

    public static void main(String[] args) {
        Adder.func();
    }
}
```

Unanswered

Question 12

0 / 1 pts

What happens if a `final` variable refers to an object?

- The variable is automatically `static`
- The reference cannot be changed to refer to a different object, but the object that is referenced can be mutated.
- The object becomes immutable.