Congratulations! You passed!

Grade received 100% To pass 80% or higher

Retake the assignment in 7h 58m

Go to next item

Objects, memory models, and scope

Latest Submission Grade 100%

1. Consider the following code: 1 / 1 point

```
public class MyClass
{
  private int a;
  public double b;

  public MyClass(int first, double second)
  {
    this.a = first;
    this.b = second;
  }
  public static void main(String[] args)
  {
    MyClass c1 = new MyClass(10, 20.5);
    MyClass c2 = new MyClass(10, 31.5);
    System.out.println(c1.a + ", " + c1.b);
  }
}
```

What would this code print?

Note that this might seem like a compiler error to be accessing c1.a and c1.b from main, but because the method main is defined in the class "MyClass", it will compile just fine (feel free to check it for yourself).

- (•) 10, 20.5
- ( ) 10, 31.5

Correct
Although "a" is a private variable, because the method "main" is in the class MyClass, it has access to that private variable. So this will run just as you'd expect.

2. 1 / 1 point

```
public class MyClass
{
  private int a;
  public double b;

  public MyClass(int first, double second)
  {
    this.a = first;
    this.b = second;
  }
  public static void main(String[] args)
  {
    MyClass c1 = new MyClass(10, 20.5);
    MyClass c2 = new MyClass(10, 31.5);
    // lines below are changed from the question above
    c2 = c1;
    c1.a = 2;
    System.out.println(c2.a);
  }
}
```

What is the result of running the code above? Hint - draw a memory model!

- (•) 2
- ( ) 10

Correct
Yes, both c1 and c2 refer to the same object. So when you change c1.a, you have effectively changed c2.a as well.

3. In the code from question 2 above, after executing c2 = c1, how can you get back the object which c2 previously pointed to (the one with a as 10 and b as 31.5)? 1 / 1 point

3. In the code from question 2 above, after executing c2 = c1, how can you get back the object which c2 previously pointed to (the one with a as 10 and b as 31.5)? 1 / 1 point

- (●) You cannot get that object back
- ( ) c2--;
- ( ) c2.restore()

Correct
Without any references pointing to that object originally referred to by c2, the garbage collector can destroy the object. If you needed to keep that object around for later, you'd need to assign it to a temporary reference.

4. Please review the code below: 1 / 1 point

```
1  public class MyClass
2  {
3    private int a;
4    public double b;
5
6    public MyClass(int first, double second)
7    {
8      this.a = first;
9      this.b = second;
10   }
11
12   // new method
13   public static void incrementBoth(MyClass c1) {
14     c1.a = c1.a + 1;
15     c1.b = c1.b + 1.0;
16   }
17
18   public static void main(String[] args)
19   {
20     MyClass c1 = new MyClass(10, 20.5);
21     MyClass c2 = new MyClass(10, 31.5);
22     // different code below
23     incrementBoth(c2);
24     System.out.println(c1.a + ", "+ c2.a);
25   }
26 }
```

What would be the output from running main?

- (●) 10, 11
- ( ) 11, 11
- ( ) 10, 10
- ( ) 11, 10

Correct
This is the correct response. c2's values are incremented by passing it as a parameter to incrementBoth.

5. Please review the code below: 1 / 1 point

```
1  public class MyClass
2  {
3    private int a;
4    public double b;
5
6    public MyClass(int first, double second)
7    {
8      this.a = first;
9      this.b = second;
10   }
11
12   public static void incrementBoth(MyClass c1) {
13     c1.a = c1.a + 1;
14     c1.b = c1.b + 1.0;
15   }
16
17   // new method
18   public static void incrementA(int first)
19   {
20     first = first + 1;
21   }
22
23   // new method
24   public static void incrementB(double second)
25   {
26     second = second + 1.0;
27   }
28
29   public static void main(String[] args)
```

```
30    {
31      MyClass c1 = new MyClass(10, 20.5);
32      MyClass c2 = new MyClass(10, 31.5);
33      // different code below
34      incrementA(c2.a);
35      incrementB(c2.b);
36      System.out.println(c2.a + ", "+ c2.b);
37    }
38  }
```

What is the output from running this code?

- (●) 10, 31.5
- ( ) 11, 32.5

Correct

This is the correct response. This is a tricky question, because methods incrementA and incrementB have no real purpose. Because Java is pass by value, a copy of that value is passed into the method. Changing the copy of the primitive type will have no impact on the parameter.