Memory Models: feedback and quiz answers

(read only after passing the quiz)

Congratulations - You're Mastering Memory Models!

Congratulations on completing the quiz on Objects, Memory Models, and Scope. The ideas in this week may have been review for your, but even so, these are key programming skills. Programmers often create memory models n their head (or on paper) when debugging and working through tricky programming projects. You'll certainly use these skills more in this course and in the later courses in the specialization.

If it took you a few tries on the quiz, don't worry about it! Memory models take practice and that's what you'll be getting as you continue with the course. Congratulations for working through the end of this week.

Quiz Answers and Explanations

Correct answers are in **bold**. Incorrect answers are shown in *italics*.

Question 1

```
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 (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.
- *10, 31.5.* This is correct if we were printing c2.a and c2.b, but we are printing c1.a and c1.b.

Question 2

```
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 (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.
- *10.* c2 now refers to the same object as c1, so the change to c1 effectively impacts c2.

Question 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)?

- *c2.restore()* There is no such generic method for a reference. This line of code would try to call a restore method on the MyClass object, which doesn't exist. Hence, this wouldn't compile.
- **You cannot get that object back (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.
- *c2--;* This is an illegal expression. The operand (c2) cannot have the unary operator '--' applied.

Question 4

Please review the code below:

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

  public MyClass(int first, double second)
  {
    this.a = first;
    this.b = second;
  }

  // new method
  public static void incrementBoth(MyClass c1) {
    c1.a = c1.a + 1;
    c1.b = c1.b + 1.0;
  }

  public static void main(String[] args)
  {
    MyClass c1 = new MyClass(10, 20.5);
    MyClass c2 = new MyClass(10, 31.5);
    // different code below
    incrementBoth(c2);
    System.out.println(c1.a + ", "+ c2.a);
  }
}
```

What would be the output from running main?

- **10, 11 (correct).** This is the correct response. c2's values are incremented by passing it as a parameter to incrementBoth.
- *11,11.* This is incorrect, the variable c1 in the scope of the method incrementBoth is different than the c1 in the scope of method main.
- *10, 10*. This is incorrect, the instance variables in c2 would be incremented by calling incrementBoth
- *11, 10.* This is incorrect, the variable c1 in the scope of the method incrementBoth is different than the c1 in the scope of method main.

Question 5

Please review the code below:

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

  public MyClass(int first, double second)
  {
    this.a = first;
    this.b = second;
  }

  public static void incrementBoth(MyClass c1) {
    c1.a = c1.a + 1;
    c1.b = c1.b + 1.0;
  }

  // new method
  public static void incrementA(int first)
  {
    first = first + 1;
  }

  // new method
  public static void incrementB(double second)
  {
    second = second + 1.0;
  }

  public static void main(String[] args)
  {
    MyClass c1 = new MyClass(10, 20.5);
    MyClass c2 = new MyClass(10, 31.5);
    // different code below
    incrementA(c2.a);
    incrementB(c2.b);
    System.out.println(c2.a + ", "+ c2.b);
  }
}
```

What is the output from running this code?

- *11, 32.5.* 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. This is a tricky question, because methods incrementA and incrementB have no real purpose.
- **10, 31.5**. 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.

✓ Complete