CSC207 Practice Quiz 1 – Winter 2017

`Last modified: Tuesday 26 September 2017`

Topics: Classes, methods, variables, constructors, static vs. instance, inheritance, types, casting, generics, memory equals vs. the `equals()` method, overriding methods, overloading constructors, Stacks, abstract classes, interfaces.

1. Define the following terms: "instance variable", "static variable", "instance method", "static method", "constructor", "primitive type" (What are the eight primitive types?), "subclasses of the `Object` class" (Are there any other classes?)

2. In the code at the end of this document, identify:

   (i) any instance variables
   (ii) any statice variables
   (iii) any instance methods
   (iv) any static methods
   (v) any constructors

3. On PCRS, you learned many properties of the `String`. Which of those properties are properties of all subclasses of `Object`? Are any of those properties specific to instances of `String` and not other classes? If so, which ones?

4. Write a main method that creates an instance of each class from the Supplementary Code, and also an instance of `LectureHall` that is of type `EventSpace`. Try calling each of the methods through each instance. When do the lookup rules apply? Does `LectureHall` inherit any variables or methods from `EventSpace`? When can you use casting to access the methods from the parent class? When you call getter methods, do they return values of variables in the same class or a different class? Write `equals` and `toString` methods for each of these classes.

5. In general, what is the `toString()` method supposed to do? What is the `equals()` method supposed to do? Why do we usually want to override them in a subclass?

6. What is type-casting?
   If ClassA is a class and ClassB extends ClassA, consider the following code:

   ```
   ClassA ob1 = new ClassA();
   ClassB ob2 = new ClassB();
   ClassA ob3 = newClassB();
   ```

   Can you think of a reason to type-cast `ob2` or `ob3` as an instance of `ClassA`? If so, what is the syntax for this? Is it possible to type-cast any of `ob1`, `ob2`, or `ob3` as an instance of `ClassB`? Why or why not?

7. `Integer` is the class associated with the primitive type `int`. When we use the `==` operator on primitive types, Java compares the values of the primitives. But for subclasses of `Object`, `==` compares the memory addresses that each instance points to. What will happen if we try to compile the following code? After fixing it so that it compiles, what will the output be?

```
int w = 5;
int x = 5
Integer y = new Integer(5);
Integer z = new Integer(5);
System.out.println(w == x);
System.out.println(w == y);
System.out.println(y == z);
System.out.println(x.equals(y)); // What is wrong with this line?
System.out.println(y.equals(z));
```

8. Consider the constructors for `EventSpace`. If we change the number of arguments in the constructor, will the code compile without also changing the constructor in `LectureHall`? If not, why not?

9. If we include a second (empty) constructor in `LectureHall`, will the code compile without also including an empty constructor in `EventSpace`? If not, why not?

10. Write a class called `TopTwoSwitch` that extends `Stack`, but keeps track of the top two items on the stack separately. It should have a method that can switch the order of the top two objects. It should have a different method that takes in two new objects as arguments and stores the previous top two objects with all of the other objects.

11. What is the difference between an abstract class and an interface? How are they similar?

12. Look up `ArrayList` on the Oracle website. Which interfaces does it implement? What is its parent class? Give three examples of useful methods contained in the `ArrayList` class. What are their arguments? What are their return types?

Supplementary Code

You can assume that all code between consecutive horizontal lines is part of a distinct file. All files are contained in the same package. With the addition of a main method, this code compiles.

```
public interface Rentable {

  public float getRentalPriceForNHours(int n);

}
```

```
public class EventSpace implements Rentable {

  private int capacity = 100;
  private float rentalPrice;
  private String location;
  public static int numRooms;

  public EventSpace(float rp, String location, boolean isRoom) {
    this.capacity = capacity;
    rentalPrice = rp;
    this.location = location;
    if(isRoom)
      numRooms++;
  }

  public int getCapactiy() {
    return capacity;
  }

  public void setCapacity(int cap) {
    this.capacity = cap;
  }

  public float getRentalPriceForNHours(int n) {
    return rentalPrice * n;
  }

  public String getLocation() {
    return location;
  }

  public void setLocation(String location) {
    this.location = location;
  }
  public static int getNumRooms() {
    return numRooms;
  }
}
```

```
public class LectureHall extends EventSpace {

  private int capacity;
  private String roomNumber;
  public static int numRooms;

  public LectureHall(int capacity, float rp, String location, boolean isRoom) {
    super(rp, location, true);
    this.roomNumber = location;
    numRooms++;
  }

  public int getCapacity() {
    return capacity;
  }

  public void setCapacity(int cap) {
    this.capacity = cap;
  }

  public String getLocation() {
    return "Room number: " + roomNumber;
  }

  public static int getNumRooms() {
    return numRooms;
  }
}
```