CSCI 1227 Sample Test

For February 2016

NOTE:

This test was put together from two other tests so as to cover the material we have seen so far this term (winter 2016). So it's actually quite a bit longer than our midterm test is going to be.

Our test will be shorter!

1. **[12]** Declare each of the following items. Use appropriate names, types, styles and modifiers. Create the most reasonable kind of item. Initialize the item if appropriate.

   a) an instance variable to keep track of how many kilometers a Car has traveled

   b) a class constant for the value of π (pi, 3.14159)

   c) a class variable to keep track of how many Student objects have been created

   d) an empty linked list of integer values

   e) an iterator over the list declared in the previous part

   f) an array of up to 150 Students (using our user-defined Student class)

2. **[4]** For each of the following method calls, say whether the method is void or value-returning. For the value-returning methods, say what the returned data type is.

   a) kbd.nextInt() // Note: kbd is a Scanner

      void value-returning: _____

   b) System.out.println() // Note: System.out is a PrintStream

      void value-returning: _____

   c) response.toUpperCase() // Note: response is a String

      void value-returning: _____

   d) Math.round(avg) // Note: avg is a double

      void value-returning: _____

3. **[6]** Suppose we have created classes for Sections, Professors, and Rooms. Suppose further than each Section has a lecturer (who is a Professor), each Professor has an office (which is a Room), and each Room has a building (which is a String). We have also created a Section object in the variable mySection. Assuming that getters have been written for each of these instance variables, which of the following commands/expressions will compile without an error message?

   a) mySection.getLecturer() OK Error

   b) mySection.getOffice() OK Error

   c) mySection.getOffice().toUpperCase() OK Error

   d) mySection.getLecturer().getOffice() OK Error

   e) mySection.getLecturer().getOffice().getBuilding() OK Error

   f) mySection.getLecturer().getOffice().toUpperCase() OK Error

4. **[4]** Suppose we have already created `drawBox(int w, int h, char edge, int indent)`, which draws a `w` by `h` box indented `indent` characters on the screen, using `edge` as the character at the edge of the box. Write *another* drawBox method, `drawBox(int w, int h)` that draws a box on the screen, indented zero characters, using a star ('*') as the character at the edge of the box. **Hint:** this is easy.

5. [4] Show the output of the following code fragment.

```
ArrayList<String> list = new ArrayList<String>();
ListIterator<String> it = list.listIterator();
it.add("Fee"); it.add("Fie");
System.out.println(list);
it.add("Foe"); it.add("Fum");
System.out.println(list);
it = list.listIterator();
it.next(); it.remove();
System.out.println(list);
it.next(); it.set("Hi");
System.out.println(list);
```

6. **[4]** Show the output of the following program. Assume that getColour and setColour do exactly what their names imply.

```
public static void main(String[] args) {
    ColourObject myObject = new ColourObject();
    ColourObject myOtherObject = new ColourObject();
    ColourObject myChosenObject = myOtherObject;

    myObject.setColour("Black");
    myOtherObject.setColour("Green");
    myChosenObject.setColour("Red");

    System.out.println(myObject.getColour());
    System.out.println(myOtherObject.getColour());
    System.out.println(myChosenObject.getColour());
}
```

7. **[4]** The following code fragment doubles all the elements of an int[] named numbers. Revise the code so that numbers is an ArrayList rather than an array. *Do not use a list iterator.*

```
for (int i = 0; i < numbers.length; ++i) {
    numbers[i] = 2 * numbers[i];
}
```

8. **[4]** Repeat the previous question *using a list iterator*.

9. **[20]** Suppose we start declaring an Amplifier class as follows:

```
public class Amplifier {
    private int volume;
}
```

   a) Write a **constructor** for the class, which takes as its one argument the initial volume for the Amplifier.

(Question 9 continues on the next page)

(Question 9, continued)

b) Write a **getter** and a **setter** for the Amplifier's volume. The setter accepts only values from 0 to 11 (inclusive). It does nothing if the value is outside that range.

c) Write a toString method for this class. The String is of this form: "Amplifier at volume 11" (where the actual volume of the amplifier is reported).

10. **[12]** Multiple Choice: select the *best available* answer from the options shown.

- A method body can contain
  a) assignment commands.
  b) input commands.
  c) method calls.
  d) output commands.
  e) (any of the above)

- A value-returning method is given information by a client program, and generates an answer. The answer should be
  a) kept private to the class.
  b) kept private to the method.
  c) printed for the user to see and returned to the client.
  d) printed for the user to see.
  e) returned to the client.

- The class `Dohickey` has two constructors: one with no parameters, and one with a single, int parameter. The declaration `new Dohickey()` creates an object equivalent to the one created by `new Dohickey(100)`. The body of the parameterless constructor would be:
  a) Dohickey = 100;
  b) Dohickey(100);
  c) this();
     Dohickey = 100;
  d) this(100);
  e) this.Dohickey = 100;

- A Thingummy object is created using the command

```
Thingummy t = new Thingummy(4, "Hello");
```

  Which of the following available constructors for the class is used?
  a) public Thingummy()
  b) public Thingummy(double d, String s)
  c) public Thingummy(int n)
  d) public Thingummy(int n, String s)
  e) public Thingummy(String s, int n)

- A variable named local is declared inside a method named doThis, in a class named Widget. The variable local can be used
  a) by any program that has a `Widget` object declared.
  b) by any program that has declared a `Widget` and called its `doThis` method.
  c) only inside the class `Widget`.
  d) only inside the method `doThis`.
  e) (any of the above, depending on whether it was declared public or private)

- Methods in the class Math (such as round, random, max, sin, cos, …) are declared
  a) private static
  b) private static final
  c) private void
  d) public static
  e) public void

- The command/expression to create a new object of the class Car would be
  a) Car myCar = new Car();
  b) Car myCar;
  c) Car new Car = myCar();
  d) new Car = myCar;
  e) new Car();

- The command/expression to create a new variable of the class Car would be
  a) Car myCar = new Car();
  b) Car myCar;
  c) Car new Car = myCar();
  d) new Car = myCar;
  e) new Car();

- The definition for the class WhatsIt has no constructor declared. This class
  a) has a default, one parameter constructor provided by Java.
  b) has a default, parameterless constructor provided by Java.
  c) will compile and have no constructors at all.
  d) will not compile because every class must have a constructor declared.
  e) will only compile if it has no instance variables, because every class with instance variables must have a constructor declared.

- Suppose that we have the following code in a program.

```
LinkedList<Integer> myLL = new LinkedList<Integer>(100);
int n = myLL.get(10);
```

What happens?

a) a) an IndexOutOfBoundsException is thrown.
b) b) the code does not compile because an Integer value can't be stored in an int variable.
c) c) the code does not compile because myLL.get(10) has not been initialized.
d) d) the variable n gets the value 0.
e) e) the variable n gets the value 100.

- Which of the following commands makes the ArrayList al empty?

a) a) al.clear();
b) b) al.empty();
c) c) al.isEmpty();
d) d) al.makeEmpty();
e) e) al.setSize(0);

- When we try to compile the following program

```
public class MultiChoice {
    public static void main(String[] args) {
        doThis();
    }

    private void doThis() {
        System.out.println("Done!");
    }
}
```

we get the error message

```
MultiChoice.java:3: non-static method doThis() cannot be
referenced from a static context
        doThis();
        ^
1 error
```

The problem is

a) the method doThis should have been declared public and static.
b) the method doThis should have been declared public.
c) the method doThis should have been declared static.
d) the method main should have been declared private.
e) the method main should **not** have been declared static.

- When there are two methods with the same name in the same class, differing only in their parameters, the method name is said to be
  a) overloaded.
  b) overrated.
  c) overruled.
  d) overwritten.
  e) (the situation described is not legal in Java)

- Suppose we intend to create a toRoman method, which converts an integer value to its Roman numeral equivalent. For example, the Roman numeral for 42 is XLII. (Note: the method builds the Roman numeral, but it does not print the Roman numeral.) Which of the following is the best stub for this method?
  a) public static int toRoman() {return 0;}
  b) public static int toRoman(String s) {return 0;}
  c) public static String toRoman(int n) {return "XLII";}
  d) public static void toRoman(int n) {}
  e) public static void toRoman(int n) {System.out.print("XLII");}

(This page is for your rough calculations. You may tear it off the exam booklet.)