Sarajevo School of Science and Technology
CS/IS 160 Programming and Problem Solving II
April 2015

Test 1 Solutions

(5% of the final grade, 90 minutes)

Problem 1 (10 points, each part 2 points).

Answer the following questions:

(a) What is the name of the function where every Java program begins execution?

Solution: main.

(b) List two primitive types in Java that store floating-point numbers.

Solution: float and double.

State whether each of the following statements is true or false. If false, explain why.

(c) The *default* case is required in the *switch* selection statement.

Solution: False. The default case in the switch selection statement is optional.

(d) A set of statements contained within a pair of parentheses is called a block.

Solution: False. A set of statements contained within a pair of curly braces is a block.

(e) Variables declared in the body of a particular method are known as *local variables*.

Solution: True.

Problem 2: Simple declarations and statements (20 points).

Write a declaration for each of the given *instance variables* below (2 point each).

(a) Declare a private Boolean variable named **isItSnowingAgain**.

Solution: `private boolean isItSnowingAgain;`

(b) Declare a public long integer variable **num** that is initialized to 42.

Solution: `public long num = 42;`

(c) Declare a private string variable **`countryName`**.

Solution: `private String contryName;`

Implement the following code snippets below (7 points each).

(d) Write a **`for`** loop that increments integer variable **`count`** from its initial value of 0 to 100 in steps of 10.

Solution: `for( int count = 0; count <= 100 ; count += 10);`

(e) Write a **`while`** loop that prints out all odd integer numbers from 0 to 100.

Solution:

```
int i = 0;
while( i <= 100 ) {
      if( i % 2 == 1 ) {
            System.out.print(i + " ");
      }
      i++;
}
```

Problem 3: Range of an integer (25 points).

The range of an integer number is defined as the difference between its minimum and maximum digit. For example, range of 2137 is (7-1) = 6, while range of 88 is 0. Also, single digit numbers have the range of 0.

Write a static method `digitRange` that returns the range of an input integer.

Use the following method prototype, where `number` is the input integer:

```
static int digitRange( int number )
```

Solution:

```
static int digitRange(int number) {

      int min = number % 10;
      int max = number % 10;
      number /= 10;

      while(number != 0) {
            int digit = number % 10;
            if( max < digit) {
                  max = digit;
            }
            else if(min > digit) {
                min = digit;
            }
            number /= 10;
        }

        return max - min;
    }
```

Problem 4: Hailstone sequence (25 points).

The *hailstone* sequence is defined as follows: Starting with a positive integer `n`, the next number in the sequence is `n/2` if `n` is even, or `3n+1` if `n` is odd. The sequence ends when it reaches 1.

For example, hailstone sequence of 6 is 6, 3, 10, 5, 16, 8, 4, 2, and 1.

Write a static method `hailstone` that takes an integer number as the starting point and prints out its full hailstone sequence to the standard output.

Use the following method prototype, where `number` is the input integer:

```
static void hailstone( int number )
```

Solution:

```
    static void hailstone(int number) {

        System.out.print(number + " ");

        while(number != 1) {
            if(number % 2 == 0) { // even case
                number /= 2;
            }
            else { // odd case
                number = 3*number + 1;
            }
            System.out.print(number + " ");
        }
        System.out.println();
    }
```

Problem 5: Circle class (20 points, each part 5 points).

We define a class named `Circle` that is a basic representation of a circle with a radius as its only instance variable. Note that the radius cannot be negative. Therefore, we need to print an error message to the console in case that the radius is set to a negative number.

We have the following Java class definition with empty constructor and methods:

```
// class definition
```

```
public class Circle {

     // instance variables
     private double radius;

     // constructor (part a)
     public Circle(double inputRadius) {}

     // getters (part b)
     public double getRadius() {}

     // public instance methods (part c and d)
     public double getArea() {}
     public double getCirc() {}

}
```

(a) Implement the constructor with an input argument for `radius`.

Solution:

```
public Circle(double inputRadius) {
   if(inputRadius < 0)
      System.out.println("Error: Radius cannot be negative!");
   else
      radius = inputRadius;
}
```

(b) Implement the getter method for the private variable `radius.`

Solution:

```
public double getRadius() {
   return radius;
}
```

(c) Implement the public instance method `getArea`, which returns the area of the circle instance. *Hint*: Area of a circle is given by $\pi \times (\text{radius})^2$.

Solution:

```
public double getArea() {
   return Math.PI*radius*radius;
}
```

(d) Implement the public instance method `getCirc`, which returns the circumference of the circle instance. *Hint*: Circumference of a circle is given by $2 \times \pi \times (\text{radius})$.

Solution:

```
public double getCirc() {
   return 2*Math.PI*radius;
}
```