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

Practice Midterm Exam Solutions

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

Answer the following questions.

(a) The number used to refer to a particular array element is called the element's _____.

Solution: index.

(b) Briefly explain what is method overloading?

Solution: Method overloading is defining methods with the same name but different number and type of arguments.

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

(c) An array is a dynamically resizable data structure.

Solution: False. Arrays in Java are fixed-size data structures.

(d) Reference-type instance variables are initialized by default to the value `null`.

Solution: True.

(e) Variables or methods declared as private are accessible only to methods of the class in which they are declared.

Solution: True.

Problem 2: Simple statements and declarations (10 points, each part 2 points).

(a) Declare and initialize a private array object `flags` that can hold 10 Boolean values.

Solution: `private boolean[] flags = new boolean[10];`

(b) Declare and initialize a public two-dimensional integer array named `intSquare` with 10 rows and 10 columns.

Solution: `public int[][] intSquare = new int[10][10];`

(c) Declare and initialize a public string array `classNames` with enough space for 100 strings.

Solution: `public String[] classNames = new String[100];`

(d) Declare a local double array `Z` that holds numbers { 1.1, 2.2, 3.14 }.

Solution: `double[] Z = {1.1, 2.2, 3.14};`

(e) Create a new integer array `Zint` to hold the same number of elements as `Z`. Populate the `Zint` array with the elements of `Z` casted to integers.

Solution: `int[] Zint = {1, 2, 3};`

Or a more general solution using a for loop:

```
int[] Zint = new int[Z.length];
for( int i=0; i < Z.length; i++ ) {
      Zint[i] = (int) Z[i];
}
```

Problem 3: Methods indexOf and lastIndexOf (20 points, each part 10 points).

(a) Implement a static method `indexOf`, which searches an integer array for a value named `key`, and returns the index of the first such element found, or -1 if none is found. Use the following method prototype:

```
static int indexOf( int[] array, int key )
```

Solution:

```
static int indexOf( int[] array, int key ) {

      for(int i=0; i < array.length; i++) {
            if( array[i] == key ) {
                  return i;
            }
      }
      return -1;
}
```

(b) Implement a static method `lastIndexOf`, which searches an integer array for a value named `key`, and returns the index of the last such element found, or -1 if none is found. Use the following method prototype:

```
static int lastIndexOf( int[] array, int key )
```

Solution:

```
static int lastIndexOf( int[] array, int key ) {

      for(int i=array.length-1; i >= 0; i--) {
            if( array[i] == key ) {
                  return i;
            }
      }
      return -1;
}
```

Problem 4: Rotate to the right (20 points).

Implement a static method `rotateRight`, which shifts the elements of an array of integers to the right one place and wraps the last element into the first place.

For example, if `array` contains elements {1, 2, 3, 4, 5}, after applying this method it would contain {5, 1, 2, 3, 4}.

Use the following method prototype:

```
static void rotateRight( int[] array )
```

Solution:

```
static void rotateRight( int[] array ) {

      int size = array.length-1;
      int lastElement = array[size];

      for(int i=size; i > 0; i--) {
            array[i] = array[i-1];
      }

      array[0] = lastElement;

}
```

Problem 5: Majority vote algorithm (20 points).

In a majority vote election, the winning candidate needs to win **more than half** of the total votes. Assume that we have two candidates in an election, denoted with numbers 1 and 2. We obtain all votes in the election as a vector of integers, and our task is to find which candidate has won the majority vote, provided that there is such a candidate. For example, the majority winner of a sequence [1 1 2 1 2 1] is element 1. As another

example, the sequence [1 1 2 2] does not have a majority winner.

Write a function called `majorityVote`, which accepts an integer array, and returns an element that is the majority winner. If there is no such element, then return 0.

Use the following function prototype:

```
static int majorityVote( int[] votes )
```

Solution:

```
static int majorityVote( int[] votes ) {

      int v1 = 0, v2 = 0;

      for( int vote : votes ) {
            if( vote == 1 ) {
                  v1++;
            }
            else if( vote == 2 ) {
                  v2++;
            }
      }

      if( v1 > votes.length/2 ) {
            return 1;
      }
      else if( v2 > votes.length/2 ) {
            return 2;
      }
      else {
            return 0;
      }
}
```

Problem 6: Complex number class (20 points, each part 5 points).

We define a class named `Complex` to implement the basic properties and operations for complex numbers. For example, 5 + 2.5*i* is a complex number, where 5 is the real part and 2.5 is the imaginary part of the number, and *i* = sqrt(-1) is the imaginary unit.

We will use two private double instance variables `real` and `imag` to represent the real and imaginary part respectively.

We have the following Java class definition:

```
// class definition
public class Complex {

     // instance variables
```

```
    private double real;
    private double imag;

    // getters and setters (part a and b)
    public double getReal() {}
    public double getImag() {}

    public void setReal(double inputReal) {}
    public void setImag(double inputImag) {}

    // public instance methods (part c and d)
    public void conjugate() {}
    public double magnitude() {}

}
```

(a) Implement the getter method for the private variable `real`.

Solution:

```
    public double getReal() {
        return real;
    }
```

(b) Implement the setter method for the private variable `real`.

Solution:

```
    public void setReal(double inputReal) {
        real = inputReal;
    }
```

(c) Implement the public instance method `conjugate`, which conjugates the complex number. For example, conjugation of complex number $3 + 4i$ will change it to $3 - 4i$ (complex conjugation flips the sign of the imaginary part).

Solution:

```
    public void conjugate() {
        imag = -imag;
    }
```

(d) Implement the public instance method `magnitude`, which returns the magnitude of the complex number. The magnitude is equal to sqrt($\text{real}^2 + \text{imag}^2$), where sqrt is the square root operation. For example, magnitude of the complex number $3 + 4i$ is equal to 5, since sqrt($3^2 + 4^2$) = 5. *Note*: You can use `Math.pow` and `Math.sqrt` static methods.

Solution:

```
    public double magnitude() {
        return Math.sqrt(real*real + imag*imag);
    }
```