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

Practice Midterm Exam

(Midterm exam: 20% of the final grade, 90 minutes)

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 _____.

(b) Briefly explain what is method overloading?

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

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

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

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

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.

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

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

(d) Declare a local double array `z` that holds numbers { 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.

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 )
```

(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 )
```

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 )
```

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 )
```

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`.

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

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

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