CSC 115
Midterm Exam:
Sections: A01 and A02
Monday, June 1st, 2020

Name:____**KEY**_____(please print clearly!)

UVic ID number:_____

Signature:_____

Exam duration: 60 minutes

Instructor: Anthony Estey

Students must check the number of pages in this examination paper before beginning to write, and report any discrepancy immediately.

- We will not answer questions during the exam. If you feel there is an error or ambiguity, write your assumption and answer the question based on that assumption.
- Answer all questions on this exam paper.
- The exam is closed book. No books or notes are permitted.
  No electronic devices of any type are permitted.
- The marks assigned to each question and to each part of a question are printed within brackets. Partial marks are available.
- There are fifteen (15) pages in this document, including this cover page.
- Page 15 is left blank for scratch work. If you write an answer on that page, clearly indicate this for the grader under the corresponding question.
- Clearly indicate only one answer to be graded. Questions with more than one answer will be given a **zero grade**.
- It is strongly recommended that you read the entire exam through from beginning to end before beginning to answer the questions.
- Please have your ID card available on the desk.

Part 1: Java Programming Fundamentals (4 marks)

1. Fill in the blanks to specify the values of the variables:

```
int x = 9;              // the value of x is 9
double a = x / 5.0;     // the value of a is 1.8
int b = x / 5;          // the value of b is 1
```

2. Fill in the value variable **x** could be initialized to so that when the code finishes execution, the final value of the result variable is 6.

```
int x = 2 ;
int result = 0;

for (int i = 0; i < 3; i++) {
    result += x;
}
```

3. Fill in the **maximum value** **x** could be initialized to so that when the code finishes execution, the final value of **`result`** is set to **2**.

```
int x = [ 6 ] ;
int y = 3;
int result = 0;
if (x == 10) {
      result = 1;
} else if ((x+y) < 10) {
      result = 2;
} else {
     result = 3;
}
```

4. Fill in the **minimum value** **x** could be initialized to so that when the code finishes execution, the final value of **`result`** is set to **3**.

```
int x = [ 5 ] ;
int y = 6;
int result = 0;
if (x == 4) {
      result = 1;
} else if ((x+y) < 10) {
      result = 2;
} else {
     result = 3;
}
```

Part 2: Arrays (8 marks)

5. Given the implementation of the sumArray method shown below, sometimes when sumArray is called, an error occurs.

   First, explain the error with the sumArray method. Then, write test code for the sumArray method that causes the error.

```
/*
 * sumArrays
 *
 * Purpose: creates an array containing the sum of values
 *          at each position in the given arrays
 *
 * Parameters: int[] a1, int[] a2 - arrays of integers
 *
 * Returns: int[] - an array of the sums at each position
 *
 * Precondition: Both a1 and a2 have been initialized
 *               and contain at least one element
 *
 * Example:
 * Given arrays: {1,2,3} and {4,5,4}, should return {5,7,7}
 */
public static int[] sumArrays(int[] a1, int[] a2) {
      int[] sumArray = new int[a1.length];
      for (int i = 0; i < a1.length; i++) {
            sumArray[i] = a1[i] + a2[i];
      }
      return sumArray;
}
```

An **error** occurs when there are more elements in a1 than a2 (a1's length is greater than a2's).

```
int[] a1 = {1, 2, 3, 4, 5};
int[] a2 = {9, 8, 7};
sumArrays(a1,a2); // throws ArrayIndexOutOfBoundsException
```

6. Does the implementation of the getMax method below return the maximum value in an array? Provide an input array that would result in the maximum value in the array returned (a 'passing' test), if you think it is possible. Provide an input array that would not result in the maximum value in the array returned (a 'failing' test), if you think it is possible.

   If you think both outcomes are possible, provide two examples and clearly indicate which produces a correct result (passing test), and which does not.

   Note: Your example arrays must contain at least 3 elements.

```
/*
 * getMax
 *
 * Purpose: finds the maximum value in the given array
 *
 * Parameters: int[] - an array of integers
 *
 * Preconditions: array contains at least three elements
 *
 * Returns: int - maximum value in the array
 */
public static int getMax(int[] arr) {
      int max = arr[0];
      for (int i = 1; i < arr.length; i++) {
            if (arr[i] > arr[i-1]) {
                  max = arr[i];
            } else {
                  max = arr[i-1];
            }
      }
      return max;
}
```

The getMax implementation always returns the maximum value of the last 2 elements in the array, so it will only return the maximum value in the array if the maximum is found in one of the final 2 positions in the array.

```
int[] a = {2, 4, 8, 10}
int[] b = {1000, 99393, 2, 1};

int result1 = getMax(a); // correct (10) value returned
int result2 = getMax(b); // incorrect (2) value returned
```

Part 3: Objects (12 marks)

```
class City {
	private String name;
	private int population;
	private String country;


	public City(String name, int population, String country) {
		this.name = name;
		this.population = population;
		this.country = country;
	}

	public String getName() {
		return name;
	}

	public int getPopulation() {
		return population;
	}

	public String getCountry() {
		return country;
	}

	public String toString() {
		return name +", "+country+". Population: "+population;
	}
}
```

7. Given the City class defined above, write code to declare a variable **c1** which is an instance of the City class with field values set to match those shown in the following table:

c1:	
name:	Vancouver
country:	Canada
population:	675,000

```
City c1 = new City("Vancouver", 675000, "Canada");
```

```
class MovieTicket {
	private String movieName;
	private double price;
	private int seatNumber;

	public MovieTicket() {
		movieName = "";
		price = 0.0;
		seatNumber = 0;
	}

	public void setMovieName(String movieName) {
		this.movieName = movieName;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public void setSeatNumber(int seatNumber) {
		this.seatNumber = seatNumber;
	}

	public String toString() {
		return "Seat number: "+seatNumber +" for " + movieName;
	}
}
```

8. Given the MovieTicket class defined above, write code to declare a variable myTicket which is an instance of the MovieTicket class with field values set to match those shown in the following table:

movieTicket:	
movieName:	Avengers
price:	13.75
seatNumber:	8

```
MovieTicket m1 = new MovieTicket();
m1.setMovieName("Avengers");
m1.setPrice(13.75);
m1.setSeatNumber(8);
```

```
class UVicCourse {
	public String subject;
	public int number;

	public UVicCourse(String subject, int number) {
		this.subject = subject;
		this.number = number;
	}

	/*
	 * compareWith
	 *
	 * Purpose: Prints out whether the current course object
	 *          has the same subject or number as values given
	 *
	 * Parameters: String subject - a course subject,
	 *                   int number - a course number
	 *
	 * Returns: nothing
	 */
	public void compareWith(String subject, int number) {
		// TODO: write code that goes here
	}
}
```

9. Given the UVicCourse class defined above, write code that would go into the compareWith method so that it works according to the documentation. An example is provided on the next page.

course1:	
subject	CSC
number	115

```
course1.compareWith("CSC", 110);
//Output:
//Same subject

course1.compareWith("MATH", 100);
//<This method call should not output anything>

course1.compareWith("ENGL", 115);
//Output:
//Same number

course1.compareWith("CSC", 115);
//Output:
//Same subject
//Same number
```

```
public void compareWith(String subject, int number) {
     if (this.subject.equals(subject)) {
          System.out.println("Same course subject");
     }
     if (this.number == number) {
          System.out.println("Same course number");
     }
}
```

```
class Student {
    private String sID;
    private int grade;

    public Student(String sID, int grade) {
        this.sID = sID;
        this.grade = grade;
    }

    public String getSID() {
        return this.sID;
    }

    public void setSID(String sID) {
        this.sID = sID;
    }

    public int getGrade() {
        return this.grade;
    }

    public void setGrade(int grade) {
        this.grade = grade;
    }

    public String toString() {
        return sID + ":" + grade;
    }

    public boolean equals(Student other) {
        return (this.sID.equals(other.sID));
    }
}
```

10. Given the Student class defined above, write the body for the static method findStudent, located in a different class, that operates on an array of Students, as shown on the next page:

```
/*
 * findStudent
 *
 * Purpose: gets the id of a student with the grade specified
 *          from the given array of students
 *
 * Parameters: Student[] classList - an array of Students
 *             int grade - the grade to search for
 *
 * Returns: String - The sID of a student with the specified grade
 *
 */
public static String findStudent(Student[] classList, int grade) {
     // TODO: write code that goes here
}
```

The following example illustrates what value is returned when implemented correctly:

studentArray:			
sID: v00123456 grade: 84	sID: v00998877 grade: 71	sID: v00335577 grade: 90	sID: v00543210 grade: 82

```
String result = findStudent(studentArray, 90);
System.out.println(result);
//Output: v00335577
```

```
public static String findStudent(Student[] classList, int grade) {
     String id = "";
     for (int i = 0; i < classList.length; i++) {
          if (classList[i].getGrade() == grade) {
               id = classList[i].getSID();
          }
     }
     return id;
}
```

Other implementation:

```
String id = "";
for (int i = 0; i < classList.length; i++) {
     if (classList[i].getGrade() == grade) {
          return classList[i].getSID();
     }
}
return id;
```

Part 4: Interfaces and ADTs (6 marks)

```
interface Animal {
      public void communicate();
}

class Cat implements Animal {
      String name;
      int age;

      public Cat(String name, int age) {
            this.name = name;
            this.age = age;
      }

      public void communicate() {
            System.out.println("Meow");
      }

      public String toString() {
            return name + " the cat is "+age+" years old";
      }
}

class Pig implements Animal {
      String name;
      int age;
      int weight;

      public Pig (String name, int age, int weight) {
            this.name = name;
            this.age=age;
            this.weight=weight;
      }

      public void communicate() {
            if (weight > 10) {
                  System.out.println("OINK OINK!!");
            } else {
                  System.out.println("oink");
            }
      }

      public String toString() {
            return name + " the pig weighs "+weight+"lbs.";
      }
}
```

11. Given the interface and classes defined on the previous page, what is the output of the code below:

```
Animal a1 = new Pig("Porky", 8, 12);
Animal a2 = new Cat("Whiskers", 4);
Animal a3 = new Pig("Penelope", 11, 9);
Animal a4 = new Cat("Garfied", 12);
Animal a5 = new Pig("Miss Piggy", 15, 2);
Animal a6 = new Pig("Pumbaa", 4, 18);

Animal[] pets = {a1, a2, a3, a4, a5, a6};

for (int i = 0; i < pets.length; i++) {
      pets[i].communicate();
}
```

```
OINK OINK!!
Meow
oink
Meow
oink
OINK OINK!!
```

```
class IntegerArrayList implements IntegerList {
        int[] data;             // array to hold list elements
        int numElements;        // number of elements in list

        // Assume the addFront, addBack, size, get, and toString methods have been
        // implemented correctly in the same way we completed them during lecture

        /* insertAt
         *
         * Purpose:  insert an item into the list at the given position
         *       and maintain the order of all other elements
         *
         * Parameters: (int) position - place to insert
         *         (int) val - value to insert
         *
         * Returns:  nothing
         *
         * Precondition: 0 <= position <= list.size()
         *
         * Example: Given a list with items [ 8 1 4 5 2 ]
         *                After calling insertAt(2, 9) the
         *                contents of the list:   [ 8 1 9 4 5 2 ]
         */
        public void insertAt (int position, int val) {
                //TODO: write code that goes here
        }
}
```

12. The class above uses an array to implement the IntegerList interface. Write code for the insertAt method. **Assume** the data array **WILL NOT** need to be resized to complete the insertion (meaning there is at least one unused position in the array).

```
public void insertAt (int position, int val) {
     for (int i = numElements; i > position; i--) {
          data[i] = data[i-1];
     }
     data[position] = val;
     numElements++;
}
```

... Left blank for scratch work...

END OF EXAM

Question	Value	Mark
Part 1	4	
Part 2	8	
Part 3	12	
Part 4	6	
Total	**30**	