COSC 1295 / 1290
Advanced Programming / Java for Programmers

Mid Semester Test, Semester 2 2016
Test 1

Instructions

1. This test is worth 10% of your final mark. However, the most valuable aspect is measuring your own progress. This test is NOT a hurdle for the course.
2. Do the test on your own. You may not refer to any online materials while you do the test.
3. This test is to be done during Week 9 lecture.
4. The allocated time for the test is **40 minutes**.
5. Write all your answers on the test paper and hand that in.

Name:

Student number:

Part A: Short Answers: 8 marks (2 marks each)

A.1 Which one of the following statements about Interface and Abstract Class is **FALSE**?

A. All methods in an abstract class must be abstract.
B. An interface cannot implement any methods, but an abstract class can.
C. An interface cannot have instance variables, but an abstract class can.
D. A Java class can extend only one direct superclass.
E. A Java class can implement more than one interface.

A -- marks
actually, B is also sort of false in Java 8 with default methods so give marks for that

A.2 Would the following code segment cause a problem? If yes, briefly explain what kind of error it would be and why. If no, briefly explain what the code would do.

```
int[] a = {1, 2, 3, 4, 5};
for (int i = 0; i < a.length; i++) {
   a[i+1] = a[i];
}
```

Yes – array index out of bounds -- 2 marks

A.3 What do each of the two "super" refer to in the following code snippet:

```
public class SportsCar extends Car
{     ...
      public SportsCar() {
        super();                            // A
      }
      ...
      public double accelerate() {
        return super.accelerate() * 1.5;    // B
      }
 }
```

A: constructor in Car / superclass

B: superclass / method in superclass
----

1 mark each

A.4 Assume that `Circle` is a subclass of `Shape`; consider the following variable declarations.

```
Shape shape1 = new Shape();
Shape shape2 = new Circle();
Circle circle2;
```

Which of the assignment statement(s) below will **NOT** cause any problem? **CIRCLE THE LETTER(S)**.

A. circle2 = (Circle) shape1;
B. circle2 = (Circle) shape2;
C. circle2 = shape1;
D. circle2 = shape2;

B – 2 marks --- 0 for anything else
A and B OR B and D OR D only – close but not quite correct (would get 1 in final exam)

PART B: Simple Programming Exercise: 8 marks

Question B.1 (8 marks)
Complete the following method that reads in an unknown number of integers from *standard input* and returns the smallest integer entered.

- You must allow for any number of input integers—the input finishes by the user typing an empty line.
- You may assume valid input—i.e. you don't have to do error-checking on the input.

```
public int min() {
        Scanner input = new Scanner( ....
        ...

        Scanner input = new Scanner(System.in);
        int i;
        int min = input.nextInt();    // should have "if hasNextInt()" but ok not to
        while (input.hasNextInt()) {
                i = input.nextInt();
                if (i < min)
                        min = i;
        }
        return min;
```

1 mark for each line above
-1 mark if min is given a default starting value (e.g. 0)

(students were given Scanner API so they should use it correctly)

PART C: Basic Object Oriented Programming: 14 marks

For this question, you need to write Java classes for a Movie Rental system.
You need a *Customer* class but **you do not have to write this class**: assume this class stores a name as a String and contains the method: *String getName().*

(a) Write the **Movie** class. **(9 marks)**
A *Movie* has a **title** (String) and a **year** (int); the year is optional. Each Movie object also keeps track of whether it is **borrowed**---it is *available* if it is not already borrowed. If a Movie is borrowed, then it stores the **Customer object** who has borrowed it.
Write the Movie class:

- Make sure you include all the right fields. You **do NOT have to write accessors** for them.
- You need two constructor to create a movie object---remember that every movie **must** have a title *when it is constructed*; a year is *optional*: a Movie can be constructed with a year, but it doesn't have to be.
- Write the method *isAvailable(),* that returns a Boolean.
- Create methods *rent(Customer c)* and *return()* -- the first method rents the movie to the Customer c and the second one "returns" a movie that is out on rent (i.e. makes it available again).
    - If a Book is not available when you try to borrow it, then it should raise an *UnavailableException* – you do **NOT** have to write this exception class: assume it already exists

```
public class Movie {
        String title;
        int year;
        Boolean available;
        Customer customer;              1 mark for all variables, ½ if ONE missing, else 0

        public Movie(String title) {
                this.title = title;             ½  mark
                available = true;               ½  mark for initialising (here or above)
        }
        public Movie(String title, int year) {  0 if no second Constructor
                this(title);                    1 mark; ½ mark if both lines above repeated
                this.year = year;                       (ok if just "this.title = title" repeated)
        }

        public boolean isAvailable() {
                return available;               1/2 mark for method correctly done (else 0)
        }
```

```
public void rent(Customer c) throws UnavailableException {  1 mark for throws
        if (available) {                                     ½ mark
                available = false;                           1 mark
                customer = c;                                1 mark
        } else {
                throw new UnavailableException();            1 mark
                                  fine if they include message in constructor
        }
}
public void return() {
        available = true;                                    1 mark
        customer = null;
}
```

(b) **Test** loop. **(5 marks)**
Our Movie rental system stores all its Movies in an ArrayList:
i) **Write an appropriate line declaring/defining this ArrayList; call it *movies***.

ii) **Write a loop** (use whatever type of loop you like) that loops through the *movies* array and prints the title of movies, and "available" if it is available OR the **name** of the Customer **for any movie that is out on loan**. Print **one entry per line**.

For example, the output may look like this:

```
Avatar: available
The Terminator: on loan to Lawrence
Titanic: on loan to Andy
Aliens: available
```

(You don't have to actually put any values into the ArrayList)

```
(i) ArrayList<Movies> movies = new ArrayList<Movies>();      1 mark
                                              Deduct 0.5 for no <Movies>

(ii)
for (int i=0; i<MAXMOVIES; i++) {                 1 mark for correct loop format
                                          for (Movie m : movies)   is OK of course
        System.out.print(movies.get(i).getTitle() + ": ");   1 mark for use of accessor
        if (movies.get(i).isAvailable()) {                   1 mark for "if "+ correct condition
                System.out.println("available");
        } else {
                System.out.println("on loan to " + movies.get(i).getCustomer().getName());   1 mark
        }
}
```