COSC 1288/1290/1295
Java for (C) Programmers

Mid Semester Test, Semester 2 2010

Instructions

1. This test is worth 5% 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 refer to the course notes if you want but you should not need to and doing so will probably slow you down.
3. This test is to be done during Week 8 lecture, except in exceptional circumstances. The allocated time for the test is **40 minutes**.
4. You must attend your Week 8 tutorial for marking and to receive feedback.
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 Any exception that can be thrown in a method *X* and is **NOT** caught inside X (Just **circle your answer**)

A. must be ignored
B. must be declared as part of the signature of the *main()* method
C. must never occur
D. must be declared as part of the method *X*'s signature
E. none of the above.

D – 2 marks; anything else is 0

A.2 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 – 2marks; anything else is 0

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
A, B, D OR B only OR D only – 1 mark
0 for anything else

PART B: Simple Programming Exercise: 8 marks

Question B.1 (8 marks)
Complete the following program 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.
- Use Scanner to read the input.

```
public class Min {
     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 will be 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. **(10 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 movie is not available when you try to rent it, then it should raise an *UnavailableException* – you do **NOT** have to write this exception!

```
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;             1 mark
                available = true;               1 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 mark for method correctly done (else 0)
        }
```

```
public void rent(Customer c) throws UnavailableException{   ½ mark for “throws ...”
        if (available) {                                    1 mark
                available = false;                          1 mark
                customer = c;                               1 mark
        } else {
                throw new UnavailableException();           ½ mark
        }
}
public void return() {
        available = true;                                   1 mark
        customer = null;
}
```

(b) **Test** loop. **(4 marks)**
Our Movie rental system stores all its Movies in an Array:

```
    Movie[] movies = new Movie[MAX_MOVIES];
```

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
```

```
for (Movie m : movies) {                                    1 mark for correct loop format
        System.out.print(m.getTitle() + “: “);              1 mark for use of accessor
        if (m.isAvailable()) {                              1 mark for “if “+ correct condition
                System.out.println(“available”);
        } else {
                System.out.println(“on loan to “ + m.getCustomer().getName());   1 mark
        }
}
```

Any loop construct is OK (if correct)
--- e.g. for (int i; i<movies.length; i++) ... then would need to use movies[i]
--- ok to not test for null
In last line, ½ only if just do m.getName() --- but full marks if getName() method defined correctly in class definition (i.e. “return customer.getName()”)