COSC 1295 / 1290
Advanced Programming / Java for Programmers

Mid Semester Test, Semester 2 2016
Test 2

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 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

A.2 Which most closely matches a description of a Java Map (e.g., a HashMap)?
(Just **circle your answer**)

A. A vector of arrays for a 2D geographic representation.
B. A class for containing unique array elements.
C. A class for containing unique vector elements.
D. An interface that ensures that implementing classes cannot contain duplicate keys.

D

A.3 Explain both instances of "this" in the following code snippet (one line each).

```
public class MyResults extends Results {
      ...
      public MyResults() {
        this(5);
      }
      public double average() {
        return this.getSum()/numberOfCourses;
      }
}
```

A: another constructor in the same class

B: this class/object; the getSum() defined method in this class

A.4 Assume that Student is a subclass of Person; consider the following variable declarations.

```
Person person1 = new Person();
Person person2 = new Student();
Student student1;
```

Which of the assignment statement(s) below will **NOT** cause any problem?
Circle your answer(s).

A. student1 = (Student) person2; OK
B. student1 = (Student) person1;
C. student1 = person2;
D. student1 = person1;

2 for A; 1 mark deducted for each incorrect one

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 mean average (i.e., the sum divided by number of integers 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 double mean() {
      Scanner input = new Scanner( ....
      ...
```

API for Scanner

- `Scanner(InputSream source)`
- `boolean hasNext()`
- `boolean hasNextLine()`
- `String next()`
- `String nextLine()`

```
Scanner input = new Scanner(System.in);
int sum = 0;
int count = 0;
while (input.hasNextInt()) {
    sum = sum + input.nextInt();
    count++;
}
return sum / count;
```

1 mark for each line + 1 mark to make 8

PART C: Basic Object Oriented Programming: 14 marks

Space to write your answers to this question is on the following pages.

For this question, you need to write Java classes for a simple university library for Books. You need a *Student* class but **you do not have to write this class**: assume this class stores an *Id* as a String and contains the method: *String getId()*.

(a) Write the **Book** class. **(9 marks)**

An entry for a *Book* has a **title** (String) and an **author** (String); the author is optional. Each Book object also keeps track of whether it is **borrowed**---it is *available* if it is not already borrowed. If a Book is borrowed, then it stores the **Student object** who has borrowed it. Write the Book class:

- Make sure you include all the right fields. You **do NOT have to write accessors** for them.
- When you write constructor/s remember that every Book **must** have a title *when it is constructed*; an author is *optional*: a Book can be constructed with or without an author.
- Write the method *isAvailable()*, that returns a boolean.
- Create methods *borrow(Student s)* and *return()* -- the first method loans the Book to the Student s and the second one "returns" a Book that is out on loan (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 Book {
        String title;
        String author;
        Boolean available;
        Student borrower;                    1 mark for all variables, ½ if ONE missing, else 0

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

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

```
public void borrow(Student s) throws UnavailableException { 1 mark for throws
        if (available) {                                          ½ mark
                available = false;                                1 mark
                student = s;                                      1 mark
        } else {
                throw new UnavailableException();          1 mark
                                        fine if they include message in constructor
        }
}
public void return() {
        available = true;                                         1 mark
        student = null;
}
```

(b) **Test** loop. **(5 marks)**

The library system stores all its Books in an ArrayList:

i) **Write the appropriate definition/declaration of this ArrayList – call it *library*.**

ii) **Write a loop** (use whatever type of loop you like) that loops through the *library* ArrayList and prints the title of books, and the **Id** of the Student **for any book that is out on loan**. Print **one entry per line**.

For example, the output may look like this:

Introduction to Java: available
Java for Winners: on loan to s12345
Advanced Java Programming: on loan to s67890
C# for Beginners: available

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

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

(ii)

```
for (int i=0; i<books.siz(); i++) {
        System.out.print(books.get(i).getTitle() + ": ");
        if (books.get(i).isAvailable()) {
                System.out.println("available");
        } else {
                System.out.println("on loan to " + books.get(i).getStudent().getName());
        }
}
```

1 mark for correct loop format
for (Movie m : movies) is OK of course
1 mark for use of accessor
1 mark for "if "+ correct condition
1 mark