Faculty of Arts & Sciences
Department of Computer Science
CMPS 200—Introduction to Computer Programming
Fall 2005–2006
Tuesday, January 24, 2006

Final Exam

Version 1

Name:______________________________ **Student Id:**______________________________

Signature:______________________________ **Section:**

Lect I	**10–11**
Lect II	**1–2**

Answers to Part I

Question	A	B	C	D	E
1.					
2.					
3.					
4.					
5.					
6.					
7.					
8.					
9.					
10.					
11.					
12.					
13.					
14.					
15.					
16.					
17.					
18.					
19.					
20.					
21.					
22.					
23.					
24.					
25.					

Grades

Part I	50	
Correct		
Incorrect		
Blank		
Part II	60	
1.1	8	
1.2	6	
1.3	6	
1.4	23	
1.5	7	
2	10	
Total	**110**	

Part II

Answer the following questions in the space provided.

Problem 1

The agencies funding the XYZ Theater Company have asked it to better keep track of the money it collects from tickets sales and maintain reduced prices for children and students thereby encouraging them to go to plays. As a friend of the theater, you volunteered to assist the Theater Company in completing the ticket sales application.

The theater company has already designed the following classes:

- `Customer` is an abstract class that defines the general properties of theater customers. This class is specialized by classes that define the properties of specific categories of customers. For example, the class `CustomerChild` specializes the `Customer` class to define the properties of child customers of the theater.
- The class `TheaterShow` represents a theater show and defines, among others, methods for accounting and seat allocation.
- The class TheaterDriver is a driver class that tests the functionality of the various other classes.

Complete the implementation of the classes below.

1. Implement the abstract class `Customer`. The class defines the instance variable `receipt` (`int`) that is initially set to -1 in the constructor. The class defines the abstract method `getDiscount()` which takes no parameters and returns a `float`. It also defines the setter method `acceptReceipt()` which sets the value of the instance variable `receipt`; it takes an int parameter and does not return a value.

(8 points = 1 +1+ 2 + 2 + 2)

```
// Class header

.............................................................................................................................................

// Variable & Constant declarations

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

// Constructor

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

// Methods

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................

.............................................................................................................................................
```

2. Implement the `CustomerChild` class which specializes the class `Customer`. The class's constructor initializes the `receipt` instance variable to –1. It also implements the method `getDiscount()` to always return the value 0.25. **(6 points = 1+1+2+2)**

```
// Class header
```

....

```
// Variable & Constant declarations
```

....

....

....

....

```
// Constructor
```

....

....

....

....

....

```
// Methods
```

....

....

....

....

....

....

....

....

....

....

3. Implement the `CustomerAdult` class which specializes the class `Customer`. The class's constructor initializes the `receipt` instance variable to –1. It also implements the method `getDiscount()` to always return the value 1.0. **(6 points = 1+1+2+2)**

```
// Class header
```

....

```
// Variable & constant declarations
```

....

....

....

....

```
// Constructor
```

....

....

....

....

....

```
// Methods
```

....

....

....

....

....

....

....

....

....

....

4. Implement the `TheaterShow` class. The class defines two instance variables: `count` (`int`) to keep track of the number of seats occupied in a particular show; `seats` (`array` of `Customer`) to store the attendees of a show. The class defines a constant TICKET_PRICE to be 10,000; this value may be discounted depending on the customer's age. The class also defines the following methods:

   - The constructor takes an `int` parameter, the number of seats in the theater. It validates this number and uses it to initialize a new Theatre Show object. An invalid parameter defaults to 100.
   - The private `issueReceipt` method returns 101 plus the number of currently allocated seats when there are available seats. It returns –1 otherwise.
   - The `addCustomer` method takes a `Customer` object as a parameter and does not return a value. It issues a receipt and, if successful, hands over the receipt to the customer (*i.e.*, the customer accepts the receipt), and assigns the next available seat to the customer. This last action updates the count of allocated seats. Upon failure, the method should issue the error message: "Sorry, the theater is already full!".
   - The `showIncome` method computes the Company's income from a show and returns it as a `float`.
   - The `toString` method returns a string describing the number of seats sold in a show along with the show's income.

(23 points = 1+1+4 +3+7+5+2)

```
// Class header
```

....

```
// Variable & constant declarations
```

....

....

....

....

```
// Constructor
```

....

....

....

....

....

....

....

....

```
// Method issueReceipt
```

```
// Method addCustomer
// Method showIncome
```

```
// Method toString
```

5. Implement the driver class `TheaterDriver`. This class's main method instantiates a `TheaterShow` object, 2 `CustomerChild` objects, 3 `CustomerStudent` objects, and 4 `CustomerAdult` objects. It then adds all these `Customer` objects to the show object and issues a report on the show's income.

(7 points)

Problem 2

An array of integers that represents a list of integers may contain repeated values. By contrast, an array of integers that represents a *set* does not contain any repetitions. **(10 points)**

Implement the `computeSetSize` static method which accepts an `int` array as a parameter. This array is full and sorted in ascending order, but may contain repeated values. The method returns the size of the array's corresponding set as an `int` value. The table below gives several examples:

Array	**Return Value**
`{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}`	`10`
`{1, 1, 1, 1, 1, 1, 1, 1, 1, 1}`	`1`
`{0, 1, 1, 1, 2, 3, 3, 4, 4}`	`5`
`{1, 1, 2, 3, 4, 4, 8, 8}`	`4`
`{1, 1, 5, 5, 7, 7, 8, 8, 8, 8}`	`4`
`{0, 1, 2, 2, 2}`	`3`
`{0, 1, 2, 3, 4, 5}`	`6`

```
public static int computeSetSize (int[] numbers)
{
}
```