**Carleton University**
**Department of Systems and Computer Engineering**
**SYSC 2004 - Object-Oriented Software Development - Winter 2015**
**Midterm Exam**

**Please read these instructions before you answer any of the exam questions:**

1. The midterm exam is closed book. Calculators are not permitted.
2. **Please, do not ask the TAs questions during the exam. Exam questions will not be explained, and no hints will be given.** If you think something is unclear or ambiguous, make a reasonable assumption (one that does not contradict the question), write it at the start of your solution, and answer the question.
3. You do not have to copy any of the Javadoc comments or Java code from the question paper to your answer booklet. You do not have to write comments in your Java code; however, feel free to add comments to clarify anything you think requires further explanation. Please don't write comments that state the obvious; for example, there is no need for comments similar to this one:

   ```
   x = x + 1;  // Add 1 to x
   ```

4. API documentation for classes `ArrayList` and `String` is provided at the end of this question paper. You probably won't need all the methods when writing your solutions; it's up to you to figure out which ones are useful and which ones aren't.
5. When you review your Java code, if you decide you need to insert additional statements, there is no need to rewrite your entire solution. Just write them below or to the side of your code, draw a box around them, and draw an arrow to indicate where they belong.
6. You can take this question paper with you when you leave the exam room.

**Overview**

There are several Web-based stores that allow customers to purchase and download eBooks; for example, Kobo or Amazon's Kindle store. A customer must first create an account with the store. Account holders can then purchase any number of eBooks. Each eBook has a title and a price. From time to time, the store offers a discount on selected eBooks; for example, eBooks by a specific author might be discounted by 10% this weekend.

Class `EBook` models eBooks. Each instance of this class stores one eBook's title, price and discount.

Account holders are modelled by class `AccountHolder`. Each instance of this class stores the name and account number of the account holder, along with a reference to the list of eBooks that this person wants to purchase; in other words, a "shopping cart".

**Question 1 [15 marks]**

Here is an incomplete implementation of class `EBook`. The specifications for the code you have to write are contained in the Javadoc comments. Read the Javadoc comments for the constructors and methods, and write the code in your answer booklet. Please write the constructor/method headers as well as their bodies. Do not copy the class header, the class' fields (instance variables) or the comments to your answer booklet. You are <u>not</u> permitted to define additional fields in this class or change the names of the fields or the signatures of the constructor and methods (e.g., parameter names and types). Your methods can, of course, define any local variables that are required. You can assume that all parameters have "sensible" values; for example, that prices are positive and discounts are in the range 0 to 100. Your code does not have to check this. **Note: marks will be deducted for "smelly" code!**

```java
/**
 * An EBook object represents an eBook's title, price and discount.
 */
public class EBook
{
    private String title;
    private double price;
    private int discount;  // 10% is stored as 10, 30% as 30, etc.
    // 0% discount by default.
    private static final int DEFAULT_DISCOUNT = 0;

    /**
     * Initialize this EBook object with the specified title and price.
     * The discount is set to the default discount.
     */
    public EBook(String title, double price) {}

    /**
     * Initialize this EBook object with the specified title, price and
     * discount.
     */
    public EBook(String title, double price, int discount) {}

    /**
     * Mutator for the discount.
     */
    public void setDiscount(int discount) {}

    /**
     * Accessor for the title.
     */
    public String getTitle() {}
```

```
    /**
     * Return this EBook's price, before any discount is applied.
     */
    public double getPrice() {}

    /**
     * Return the price that the account holder will pay for this
     * EBook, after any discount has been applied.
     */
    public double getDiscountedPrice() {}
}
```

**Question 2 [15 marks]**

Here is an incomplete implementation of class `AccountHolder`. The specifications for the code you have to write are contained in the Javadoc comments. Read the Javadoc comments for the constructors and methods, and write the code in your answer booklet. Please write the constructor/method headers as well as their bodies. Do not copy the class header, the fields or the comments to your answer booklet. You are not permitted to define additional fields in this class or change the signatures of the constructor and methods (e.g., parameter names and types). Your methods can, of course, define any local variables that are required. **Note: marks will be deducted for "smelly" code!**

```
import java.util.*;

/**
 * An account holder has a name, an account number, and a list of the
 * EBooks that the person is going to purchase (a "shopping cart").
 */
public class AccountHolder
{
    private String name;
    private int accountNumber;
    private ArrayList<EBook> shoppingCart;

    /**
     * Initialize this AccountHolder with the specified name and
     * account number, and create an empty list for the shopping cart.
     */
    public AccountHolder(String name, int accNum) {}

    /**
     * Empty this AccountHolder's shopping cart, discarding any EBooks
     * any eBooks that were in the cart before this method was called.
     */
     public void emptyShoppingCart() {}
```

```
    /**
     * Add the EBook referred to by parameter book to the
     * shopping cart, according to these rules:
     * - if an EBook with the same title is already in the
     *   shopping cart, replace it with book.
     * - if no EBook with the same title is currently in the
     *   shopping cart, add book to the cart.
     */
    public void addEBook(EBook book) {}

    /**
     * Return the count of the number of EBooks in the
     * shoppingcart.
     */
    public int sizeOfCart() {}

    /**
     * Return the purchase price of all the EBooks in the
     * shopping cart. This price includes all discounts.
     */
    public double totalPurchasePrice() {}
}
```

**Question 3 [7 marks]**

A preferred account holder is a kind of account holder who receives an extra discount if they purchase more than 5 books at one time. It should be possible to change the extra discount at any time. This type of account holder is modelled by class `PreferredAccountHolder`.

When answering this question, you are not permitted to make any changes to the `AccountHolder` class from Question 2; i.e., you cannot define additional fields or methods in `AccountHolder`.

Write the complete class definition for `PreferredAccountHolder`. This class must be a subclass of `AccountHolder`. You can assume that all parameters have "sensible" values; for example, that discounts are in the range 0 to 100. **Note: marks will be deducted for "smelly" code!**

- Define a constructor to initialize the account holder's name, account number and extra discount, and perform any other required initialization;
- Define a `totalPurchasePrice` method that returns the purchase price of all the eBooks in the shopping cart after the extra discount, if applicable, has been applied. This method's signature is:

```
public double totalPurchasePrice()
```

Example: suppose a `PreferredAccountHolder` receives an extra discount of 10% if more than 5 books are purchased. Suppose 7 books are in the shopping cart, and the `totalPurchasePrice` method in class `AccountHolder` returns 50.0 ($50.00). The `totalPurchasePrice` method in `PreferredAccountHolder` should return 45.0 ($45.00); that is, $50.00 minus an extra 10% discount.

- Define any new methods or override any inherited methods, as required.

**Question 4 [3 marks]**

Suppose you prepared a BlueJ project containing your `EBook`, `AccountHolder` and `PreferredAccountHolder` classes. Draw a class diagram showing how this project would be represented pictorially in BlueJ's main window.

**Question 5 [5 marks]**

Write the Java statements that create a `PreferredAccountHolder` object for J. Smith, whose account number is 123456. This account holder receives an extra discount of 20%. The reference to the object should be assigned to a variable named `myAccount`. Don't write a complete class or method; just write the required statements.

Suppose the newly created object has been placed on BlueJ's object bench. We open an object inspector to verify that the object was initialized correctly. Draw a picture of the object inspector; i.e., what fields appear in the inspector, and what is in each field?

## API Summary - Class `ArrayList<E>`

```
// Appends o (of type E) to the end of this list. Returns true if
// successful, otherwise returns false.

boolean add(E o);

// Removes all the objects from the list. The list will be empty when
// this method returns.

void clear();

// Returns the object (of type E) at the specified position (index)
// in the list. index must be >= 0 and < size().
// The object is not removed from the list.

E get(int index);

// Returns true if this list has no objects, otherwise returns false.

boolean isEmpty();

// Removes the object (of type E) at the specified position (index)
// in the list. index must be >= 0 and < size(). Shifts any subsequent
// elements to the left (subtracts one from their indices).
// Returns the object that was removed from the list.

E remove(int index);

// Replaces the object (of type E) at the specified position (index) in
// this list with the specified element. index must be >=0 and < size().
// Returns the object that was previously stored at the specified
// position.

E set(int index, E element);

// Returns the number of objects stored in this list.

int size();
```

## API Summary – Class `String`

```
// Compares this string with the object provided as the argument
// Returns true if the strings have the same content

boolean equals(Object anObject);
```