Assignment 5:
This assignment consists of four parts: A, B, C and D.
Deadline: **27 February, 11:59 pm**

Part A:

Class Mreview (in "Mreview.java")

Mreview is a class which contains a movie title and the movie's ratings. Ratings are stored in a Java ArrayList<Integer> object.

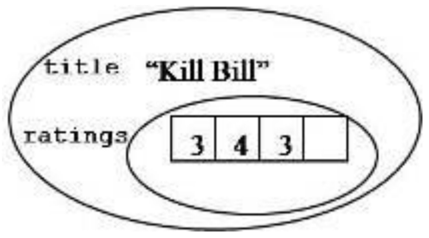
This class should have 2 member variables: title and ratings.

```
import java.util.*;

public class Mreview implements Comparable<Mreview>
{
  // instance variables
  private String title;   // title of the movie
  private ArrayList<Integer> ratings; // list of ratings stored in a
Store object

  // methods
  ...
}
```

- Conceptually, an Mreview object whose title is "Kill Bill" and has ratings 3, 4, 3 looks like:



- Your job is to write **a total of 10 methods** (3 constructors + 7 other methods) to the class AND write a test code in main().
- Read the **javadoc for Mreview** (http://condor.depaul.edu/ntomuro/courses/301/assign/hw1files/Mreview.html) for the specifics of the methods.
- For main(), you create at least 1 Mreview object and add 3 ratings, then print the object by calling the toString() method and its average rating by calling the aveRating() method.

Part B:

Implement the following two classes which are in an inheritance relation:

```
     Account
       |
  CheckingAccount
```

Your job is to complete the classes.

1. Class Account (partially filled **"Account.java"**)

- Class Account is a superclass of CheckingAccount and SavingsAccount.
- The class have the following three protected data members: FirstName (a String), LastName (a String), and CurBalance (a double).
- Following are the public methods that this class provides.
    1. public Account(String fname, String lname, double curbalance) // constructor with parameters

2. public String getAcctType()
   -- Returns a String of the class name "Account" (and that's all it does).
3. **public double DebitTransaction(double debitAmount)**
   -- Subtracts the transaction debit amount from the current balance, and returns the new balance. No condition check is required for subtracting.
4. **public double CreditTransaction(double creditAmount)**
   -- Adds the transaction credit amount to the current balance, and returns the new balance.
5. public String toString()
   --- Returns a String representation of the class, in the form "Account name: .., Account Type: ..., Balance $..".

2. Class CheckingAccount (partially filled "CheckingAccount.java")

This account type has to have a pre-set minimum balance. For every transaction resulting in (i.e., AFTER the transaction) the amount less than the minimum balance, a pre-set penalty fee is charged. The balance can become negative. Following are the methods that this class provides:

1. **public CheckingAccount(String fname, String lname, double cb)**
   -- The three parameters should be utilized to initialize the members inherited from the base class Account.
   -- Call the super class Account's constructor to do so.
2. **public double DebitTransaction(double debitAmount)**
   -- This method overrides the method with the same name inherited from the super class Account.
   -- First call the super class' method (with the given parameter). Then call ChargeFee() of this class to possibly

charge a fee (if the resulting balance went down below the MinBalance).
-- Then returns the resulting CurBalance.

3. **public double CreditTransaction(double creditAmount)**
-- This method overrides the method with the same name inherited from the super class Account.
-- First call the super class' method (with the given parameter). Then call ChargeFee() of this class to possibly charge a fee (if the resulting balance is still below the MinBalance).Adds the transaction amount to the account and possibly charges a transaction fee.
-- Then returns the resulting CurBalance.
4. **private void ChargeFee()**
-- This is a private method. It is called internally by the debit and credit transaction methods.
-- This method subtracts Fee from CurBalance if the current CurBalance is below (<) MinBalance.

Part C:

This assignment requires you to develop an object oriented software system in Java that will keep track of pets treated and boarded in an animal hospital. Detail class specifications (data members, methods and access modifiers) are described below. Please note the following requirements:

• You need to implement each class in a separate file.
• While implementing the design you may want to follow the order the classes are specified below and test each class individually.

Class: Pet (File name: Pet.java)

The class should have the following three private data members, Pet name (a String), owner name (a String), and color (a String), and one protected data members for sex (an integer, but it will only hold one of the following four public static final int values: MALE, FEMALE, SPAYED and NEUTERED. You should define these four static finals in your class).

Following are the public methods that this class should provide:
Pet (String name, String ownerName, String color); //Constructor
String getPetName();
String getOwnerName();
String getColor();
void setSex(int sexid);
String getSex(); // Should return the string equivalent of the gender, e.g the string "MALE" etc.
String toString(); // Should return the name, owner's name, age, color, and gender (use getSex());

A Sample (preferred) return value by toString is as follows:

Spot owned by Mary
Color: Black and White
Sex: Male

Interface: Boardable (File name: Boardable.java)

This interface, should include the following public methods:

void setBoardStart(int month, int day, int year);
void setBoardEnd(int month, int day, int year);
boolean boarding(int month, int day, int year);

See the Cat and Dog classes for what these methods should do when implemented. Note, the month will be in the range 1-12, day in the range 1-31, and year will be a four digit number.

Class: Cat (File name: Cat.java)

This class should extend the Pet class and implement the Boardable interface. In addition to the data members and methods inherited from Pet, the Cat class should have a private hairLength data member, which is a string. Following are the public methods that this class should provide Cat (String name, String ownerName, String color, String hairLength);
// Do not forget to call super.
String getHairLength(); // returns the string hairLength
String toString()
/* method that returns a String that identifies the pet as Cat and returns a complete description of the cat, including the values stored in the Pet parent class.*/

A Sample (preferred) return value by toString is as follows:

```
CAT:
Tom owned by Bob
Color: black
Sex: spayed
Hair: short
```

In order to implement the Boardable interface define new data members to store the boarding start and end dates, implement the setBoardStart and setBoardEnd methods to store values for these data members. Also implement the boarding method to return true if the given data is between the start and end dates, otherwise it returns false. Note: You should also return true if the given date is equal to the start or end date.

Class: Dog (File name: Dog.java)

This class should extend the Pet class and implement the Boardable interface. In addition to the data members and methods inherited from Pet, the Dog class should have a private size data member, which is a string. Following are the public methods that this class should provide:
Dog (String name, String ownerName, String color, String size);
// Constructor must set the size. Do not forget to call super.
String getSize(); // returns the string size String toString();
/* method that returns a String that identifies the pet as Dog and returns a complete description of the dog, including the values stored in the Pet parent class. */

A Sample (preferred) return value by toString is as follows:

DOG:
Spot owned by Susan
Color: white
Sex: spayed
Size: medium

In order to implement the Boardable interface define new data members to store the boarding start and end dates, implement the setBoardStart and setBoardEnd methods to store values for these data members. Also implement the boarding method to return true if the given data is between the start and end dates, otherwise it returns false. Note: You should also return true if the given date is equal to the start or end date.

Part D:

Given a target integer target and an integer array A sorted in ascending order, find the index i in A such that A[i] is closest to target. Please write main() method to test your code.

Assumptions:

1. There can be duplicate elements in the array, and we can return any of the indices with same value.
2. if A is null or A is empty, return -1.

Example:
A = [1, 3, 3, 4], target = 2, return 0 or 1 or 2
A = [0, 1, 5], target = 7, return 2