York University

AK/ITEC 1620 3.0 – Section A
OBJECT-BASED PROGRAMMING

Fall 2008
Final Exam

Examiner: S. Chen
Duration: Three Hours

This exam is closed textbook(s) and closed notes. Use of any electronic device (e.g. for computing and/or communicating) is NOT permitted.

Do not unstaple this test book – any detached sheets will be discarded. Answer all questions in the space provided. No additional sheets are permitted.

Work independently. The value of each part of each question is indicated. The total value of all questions is 100.

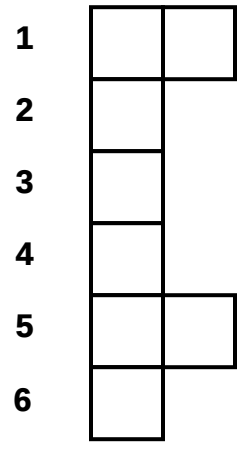
Write your name and student number in the space below, and on the top of each sheet of this exam where indicated.

NOTE: YOU MAY USE PEN OR PENCIL.

Surname: _____

Given Names: _____

Student Number: _____



Total

Question 1 (15 marks) Object Diagrams:
Answer both parts below.

The file `One.java` contains the following implementation of the `One` class:

```
public class One
{
     public int data;

     public One (int data)
     {
     }
}
```

The file `Two.java` contains the following implementation of the `Two` class:

```
public class Two
{
     public One obj;
     public int data;

     public Two (One obj, int data)
     {
     }
}
```

The `main` method in the file `MainClass.java` uses the above classes:

```
public class MainClass
{
     public static void main(String[] args)
     {
          One x = new One(5);
          One y = new One(5);
          Two a = new Two(new One(3), 7);

          // Part 1 – draw the object diagrams at this time

          x.data = 1;
          a.data = y.data;
          y = x;
          a.obj = x;
          x = new One(2);

          // Part 2 – draw the object diagrams at this time
     }
}
```

Surname:_____ First name:_____ Student #: _____

When `java MainClass` is executed,

Part 1 (**5 marks**): draw the object diagrams for all identifiers of `One` and `Two` when the comment line "`// Part 1 – draw the object diagrams at this time`" is reached.

Part 2 (**10 marks**): draw the object diagrams for all identifiers of `One` and `Two` when the comment line "`// Part 2 – draw the object diagrams at this time`" is reached.

Question 2 (15 marks) Structured Programming:

Write a program in JAVA that will simulate coin tosses and output the number of tosses required to get three "heads" in a row. This program will also output to the screen the "heads" and "tails" that it has simulated tossing.

For example, your program may produce output as follows:

Example 1:

```
HTHHH
5
```

Example 2:

```
TTTHTHHTTTTHHH
14
```

Please write your program on the following page.
You may use this page for rough work, but <u>anything on this page will not be graded</u>.

Surname:_____ First name:_____ Student #: _____

```
import java.until.*;

public class Question2
{
  public static void main (String[] args)
  {









  }
}
```

Question 3 (15 marks) Arrays:

An array (which is fully populated) is being used as a circular buffer. Write a code fragment in JAVA that will shuffle all elements of the array back one, and move the last element to the front of the rotated array.

Examples are given below (note: there are not outputs):

Example 1:

```
int[] array = new int[] {1, 2, 3};
```

array will become {3, 1, 2}

Example 2:

```
int[] array = new int[] {1, 2, 3, 2};
```

array will become {2, 1, 2, 3}

Example 3:

```
int[] array = new int[] {-1, 0, 3};
```

array will become {3, -1, 0}

Please write your code on the following page.
You may use this page for rough work, but <u>anything on this page will not be graded</u>.

Surname:_____ First name:_____ Student #: _____

```
// int[] array;
```

Question 4 (15 marks) Arrays of Objects:
The API for the `Card` class is given below. Each instance of this class represents a playing card that can be found in a standard deck of 52 cards.

Field Summary	
`static int`	`ACE` The int value representing an Ace.
`static int`	`CLUBS` The int value representing the Clubs suit.
`static int`	`DIAMONDS` The int value representing the Diamonds suit.
`static int`	`HEARTS` The int value representing the Hearts suit.
`static int`	`JACK` The int value representing a Jack.
`static int`	`KING` The int value representing a King.
`static int`	`QUEEN` The int value representing a Queen.
`static int`	`SPADES` The int value representing the Spades suit.
Constructor Summary	
`Card (int suit, int value)` Constructs a new Card with the given `suit` and `value`. If the suit is invalid, the new Card will be a Spade, and if the value is invalid, the new Card will be an Ace.	
Method Summary	
`int`	`getSuit ()` Returns the suit of this Card. The possible values are CLUBS, DIAMONDS, HEARTS, and SPADES.
`int`	`getValue ()` Returns the value of this Card. The possible values are 2-10 for the numbered Cards, ACE, KING, QUEEN, and JACK.

Surname:_____ First name:_____ Student #: _____

Write a code fragment in JAVA that will output “One Suit” if all the Cards in a hand are the same suit, and “Mixed Suits” otherwise. (You may assume the hand has at least 2 Cards and is fully populated.)

```
// Card[] hand;
```

Question 5 (25 marks) Object-Based Programming:
Answer both parts below.

The API for the `CallTimer` class is given below. An instance of this class can report on the amount of time that occurs between the most recent calls to start() and stop().

Constructor Summary	
`CallTimer ()` Constructs a new Call Timer.	
Method Summary	
`Time`	`getTime ()` Returns the amount of Time between start() and stop().
`void`	`start ()` Starts the Call Timer.
`void`	`stop ()` Stops the Call Timer.

The API for the `TimeLog` class is given below. Each instance of this class stores an amount of Time that can be used for record keeping.

Constructor Summary	
`TimeLog ()` Constructs a new Time Log with an initial amount of zero Time.	
Method Summary	
`Time`	`getTime ()` Returns the amount of Time currently logged.
`void`	`resetTime ()` Resets the amount of Time currently logged to zero.
`void`	`setTime (Time updateTime)` Sets the amount of Time currently logged to the updated Time.

The API for the `Time` class is given below. Each instance of this class represents an amount of minutes and seconds. The amount seconds will be an integer between 0 and 59 (inclusive), and the amount of minutes will never be negative.

Field Summary	
`static int`	`MINUTES` The number of minutes in an hour.
`static int`	`SECONDS` The number of seconds in a minute.
Constructor Summary	
`Time()` Constructs a new Time amount with 0 minutes, and 0 seconds.	
`Time(int m, int s)` Constructs a new Time amount with `m` minutes and `s` seconds. Note: if `h`, `m`, or `s` are invalid, a run-time error will occur.	
Method Summary	
`Time`	`add(Time amount)` Returns a Time that is the sum of this Time and the given `amount`.
`void`	`addTime(Time amount)` Increases this Time by the given `amount`.
`int`	`getMinutes ()` Returns the number of minutes in this Time.
`int`	`getSeconds ()` Returns the number of seconds in this Time.
`boolean`	`isGreaterOrEquals(Time amount)` Returns `true` if this Time amount is greater than of equal to `amount`.
`boolean`	`isEquals(Time amount)` Returns `true` if this Time amount is equal to `amount`.
`Time`	`subtract(Time amount)` Returns a Time that is the difference of this Time and the given `amount`.
`void`	`subtractTime(Time amount)` Decreases this Time by the given `amount`.

Surname:_____ First name:_____ Student #: _____

Part 1 (**10 marks**):

A certain phone company charges all partial minutes as full minutes (e.g. 3 minutes 27 seconds and 4 minutes 0 seconds are both charged as 4 minutes). Assuming that the available CallTimer has just finished timing a phone call, create a new instance of Time that represents the Time to be billed (e.g. 4 minutes 0 second in the previous example).

```
// CallTimer timer;
// Time billedTime;
```

Surname:_____ First name:_____ Student #: _____

Part 2 (**15 marks**):

To keep track of your **total** phone usage, a TimeLog can be used. For a new call recorded by the given CallTimer, update the existing TimeLog so that it stores the total Time used for phone calls.

```
// TimeLog log;
// CallTimer timer;
```

Question 6 (15 marks) Collections:

The API for the `Money` class is given below. Each instance of this class represents an amount of dollars and cents. The amount of cents will be an integer between 0 and 99 (inclusive), and the amount of dollars will never be less than zero.

Constructor Summary	
`Money ()` Constructs a new Money amount with 0 dollars and 0 cents.	
Method Summary	
`void`	`add (Money amount)` Adds the given `amount` of Money to this amount
`boolean`	`equals (Money amount)` Returns `true` if this Money amount is equal to `amount`.
`boolean`	`isGreaterOrEquals (Money amount)` Returns `true` if this Money amount is greater than or equal to `amount`.
`Money`	`multiply (int multiple)` Returns a new Money amount that is equal to this amount times the given `multiple`.

The API for the `Transaction` class is given below. Each instance of this class represents a deposit or a withdrawal for an amount of Money.

Field Summary	
`static int`	`DEPOSIT` The int code for a deposit.
`static int`	`WITHDRAWAL` The int code for a withdrawal.
Method Summary	
`Money`	`getAmount ()` Returns the amount of Money for this Transaction.
`int`	`getType ()` Returns the int code for this Transaction.

The API for the `BankStatement` class is given below. Each instance of this class represents all of the transactions that have been made in a given month.

Constructor Summary	
`BankStatement ()` Constructs a new BankStatement with a starting balance of zero.	
`BankStatement (Money startBalance)` Constructs a new BankStatement with the given `startBalance`.	
Method Summary	
`void`	`addTransaction (Transaction action)` Adds the given `action` to the list of Transactions recorded by this BankStatement.
`void`	`cancelTransaction (int i)` Removes the `i`th Transaction from this BankStatement. After removal, the `i` +1 Transaction (if it exists) will be listed as the `i`th Transaction.
`Money`	`getCurrentBalance ()` Returns the current balance after all Transactions have been processed.
`int`	`getNumberOfTransactions ()` Returns the total number of Transactions processed during this BankStatemtnt
`Money`	`getStartBalance ()` Returns the starting balance for this BankStatement.
`Transaction`	`getTransaction (int i)` Returns the `i`th Transaction of this BankStatement. Transactions are numbered from 0 to n-1 where n is the total number of Transactions. An index `i` outside of this range will cause the method to return `null`.

Surname:_____ First name:_____ Student #: _____

Write a code fragment in JAVA that will determine the total value of deposits made during the given Bank Statement.

```
// Money totalValue;
// BankStatement statement;
```