York University

AK/ITEC 1620 3.0
OBJECT-BASED PROGRAMMING

Final Exam Sample

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: _____

1	
2	
3	
4	
5	
6	

Total

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

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

```
public class First
{
     public int value;

     public First (int value)
     {
     }
}
```

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

```
public class Second
{
     public First first;
     public int second;

     public Second (First first, int second)
     {
     }
}
```

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

```
public class MainClass
{
     public static void main(String[] args)
     {
          First x = new First(8);
          First y = new First(1);
          Second a = new Second(y, 2);

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

          y = x;
          a.second = a.first.value;
          x = new First(7);
          a.first = y;

          // 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 `First` and `Second` 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 `First` and `Second` 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 determine if the input int value represents a valid Wizard Ball score. In Wizard Ball, there are two ways to score – "lightning bolts" which are worth 7 points and "fire balls" which are worth 11 points. To be a valid score, it must be possible to achieve the given score with an integer number of lightning bolts and fire balls.

For example, the following inputs would lead to the underlined output:

Example 1:

28
valid // four lightning bolts, zero fire balls

Example 2:

36
valid // two lightning bolts, two fire balls

Example 3:

19
not valid // no possible combinations

Example 4:

0
valid // zero lightning bolts, zero fire balls

Example 5:

12
not valid // no possible combinations

Example 6:

22
valid // zero lightning bolts, two fire balls

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

Surname:_____ First name:_____ Student #: _____

```
import java.until.*;

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









  }
}
```

Question 3 (15 marks) Arrays:

Write a code fragement in JAVA that will determine if a fully populated array of ints has more odd values than even values. Only consider positive values – all values less than or equal to zero will not be counted as being either even or odd.

Example outputs (underlined) are given below:

Example 1:

```
int[] array1 = new int[] {1, 2, 3};       // two odd values, one even value
```

Array has more odd

Example 2:

```
int[] array1 = new int[] {1, 2, 3, 2};   // two odd values, two even values
```

(no output)

Example 3:

```
int[] array1 = new int[] {-1, 2, 3};      // one odd value, one even value,
                                             one non-positive value
```

(no output)

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

Surname:_____ First name:_____ Student #: _____

```
// int[] array1;
```

Question 4 (15 marks) Arrays of Objects:

The API for the `DigitCode` class is given below. Each instance of this class represents a two-digit code. Each digit is an integer from 0-9 (inclusive), and a code must contain two distinct digits to be valid – e.g. the codes 0,0 and 8,8 are invalid because the first and second digits in both codes are the same. The default code is 1,2 – 1 is the first digit of the code, and 2 is the second digit of the code.

<table>
<tr><td colspan="2">Field Summary</td></tr>
<tr><td><code>static int</code></td><td><code>DEFAULT_FIRST</code>
The first digit of the default code.</td></tr>
<tr><td><code>static int</code></td><td><code>DEFAULT_SECOND</code>
The second digit of the default code.</td></tr>
<tr><td colspan="2">Constructor Summary</td></tr>
<tr><td colspan="2"><code>DigitCode ()</code>
Constructs a new two-digit code set to the default – 1,2.</td></tr>
<tr><td colspan="2">Method Summary</td></tr>
<tr><td><code>boolean</code></td><td><code>changeCode (int old1, int old2,
int new1, int new2)</code>
Attempts to change the code to <code>new1</code>,<code>new2</code>. To do so, <code>old1</code>,<code>old2</code> must match the current code, and <code>new1</code>,<code>new2</code> must represent a valid code. Returns <code>true</code> if the code is changed, and <code>false</code> otherwise.

Example: if the current code is 1,2, then <code>changeCode(1,2,3,4)</code> will change the code to 3,4 and return <code>true</code>. If the current code is 3,4, then <code>changeCode(3,4,5,5)</code> will leave the code unchanged and return <code>false</code>.</td></tr>
<tr><td><code>boolean</code></td><td><code>isCode (int first, int second)</code>
Returns <code>true</code> if the code is <code>first</code>,<code>second</code>, and <code>false</code> otherwise.

Example: if the current code is 1,2, then <code>isCode(3,4)</code> will return <code>false</code>.</td></tr>
</table>

Given a partially-populated array of DigitCodes where each DigitCode has the default code, write a code fragment in JAVA that will set each DigitCode to a random code that is not the default code.

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

Surname:_____ First name:_____ Student #: _____

```
// DigitCode[] codes;
// int count;
```

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

The API for the `RiderCard` class is given below. Each instance of this class stores the fare category of the rider.

Field Summary	
`static int`	`ADULT` The value that represents an adult rider.
`static int`	`SENIOR` The value that represents a senior rider.
`static int`	`STUDENT` The value that represents a student rider.
Constructor Summary	
`RiderCard (int category)` Constructs a new RiderCard with the given fare `category` for the rider.	
Method Summary	
`int`	`getCategory ()` Returns the fare category for this RiderCard.

The API for the `FareGuide` class is given below. The methods allow the number of fare units or stops travelled to be converted into the other based on the category of the rider.

Method Summary	
`static int`	`getMaxStops (RiderCard rider, FareCard fare)` Returns the maximum number of stops that the `rider` can travel on the fare units available on the given FareCard.
`static int`	`getFare (RiderCard rider, int stops)` Returns the number of fare units that the `rider` requires to travel the given number of `stops`.

The API for the `FareCard` class is given below. Each instance of this class will keep the number of fare units available for use on that card.

Field Summary	
`static int`	`MAX_UNITS` The maximum number of fare units that a FareCard can store.
Constructor Summary	
`FareCard ()` Constructs a new FareCard with 0 fare units.	
`FareCard (int units)` Constructs a new FareCard. If the given number of `units` is allowable (i.e. between 0 and the maximum allowed), the initial number of units will be set to this value. Otherwise, the FareCard will start with the maximum number of units allowed.	
Method Summary	
`int`	`getUnits ()` Returns the number of units on this FareCard.
`boolean`	`useUnits (int units)` If the number of units remaining on this FareCard is greater than or equal to 0, then the method returns `true` and decreases the number of units on this FareCard by `units`. Otherwise, the number of units remains unchanged, and the method returns `false`.
`int`	`buyMaxUnits ()` Increases the number of fare units on this FareCard to the maximum allowed, and returns the number of fare units that were purchased to do so.
`void`	`buyUnits (int units)` Increases the number of units on this FareCard by `units`. If the number of units on this FareCard becomes greater than the maximum allowed, then this FareCard will store the maximum number of units allowed.
`int`	`ridesAvailable (int units)` Returns the number of rides available with this FareCard if each ride requires the given number of `units`.

Surname:_____ First name:_____ Student #: _____

Part 1 (**10 marks**):

Write a code fragment in JAVA that will create a new FareCard that contains exactly enough fare units for the given rider to travel the given number of stops.

```
// FareCard newFareCard;
// RiderCard rider
// int stops
```

Surname:_____ First name:_____ Student #: _____

Part 2 (**15 marks**):

Write a code fragment in JAVA that will determine if the given FareCard has enough fare units to allow a rider with the given fare category to travel the given number of stops. Set the variable enoughFare to `true` if the FareCard has enough fare units, and to `false` otherwise.

```
// boolean enoughFare;
// FareCard card
// int category
// int stops
```

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 `Stock` class is given below. Each instance contains information for a stock issue.

Constructor Summary	
`Stock (String name, Money price)` Constructs a new Stock with the given `name` and `price`.	
Method Summary	
`String`	`getName ()` Returns the name of this Stock.
`Money`	`getPrice ()` Returns the price of this Stock.
`void`	`setPrice (Money price)` Updates the price of this Stock.

The API for the `Portfolio` class is given below. Each instance of this class represents an investment Portfolio that can consist of stocks and cash.

Field Summary	
`static int`	`MAX_STOCKS` The maximum number of Stocks that can be held in a Portfolio.
Constructor Summary	
`Portfolio ()` Constructs a new Portfolio with no Stocks and no Money.	
`Portfolio (Money cash)` Constructs a new Portfolio with no Stocks and the given amount of `cash`.	
Method Summary	
`void`	`addCash (Money amount)` Adds the given `amount` of Money to this Portfolio.
`void`	`addStock (Stock stock, int quantity)` Adds the given Stock with the given quantity to this Portfolio.
`int`	`find (String name)` Returns the index for the given Stock. Returns -1 if the given Stock is not in this Portfolio.
`Money`	`getCash ()` Returns the amount of Cash currently in this Portfolio.
`int`	`getNumberOfStocks ()` Returns the number of (different) Stocks currently held in this Portfolio.
`int`	`getQuantity (int i)` Returns the number of shares owned for the i^{th} Stock in this Portfolio. Stocks are 0-indexed (from 0 to the number of stocks – 1).
`Stock`	`getStock (int i)` Returns the i^{th} Stock in this Portfolio. Stocks are 0-indexed (from 0 to the number of stocks – 1).
`void`	`withdraw (Money amount)` Withdraws the given `amount` of Money from this Portfolio.

Surname:_____ First name:_____ Student #: _____

Write a code fragment in JAVA that will determine the total value of the given Portfolio.

```
// Money totalValue
// Portfolio investments
```