Université d'Ottawa
Faculté de génie

École d'ingénierie et de
technologie de l'information

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Information
Technology and Engineering

Introduction to Computing II (ITI 1121)
Midterm Examination

Instructor: Marcel Turcotte

March 2013, duration: 2 hours

Identification

Student name: _____

Student number: _____ Signature: _____

Instructions

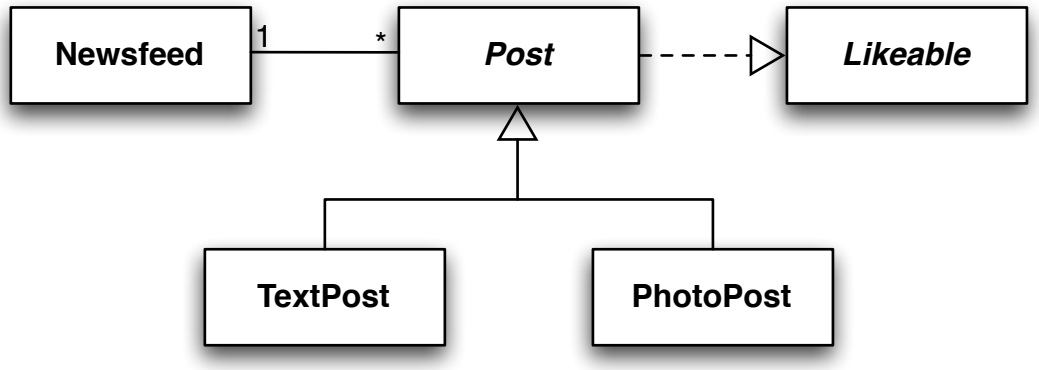
1. This is a closed book examination;
2. No calculators, electronic devices or other aids are permitted;
   (a) Any electronic device or tool must be shut off, stored and out of reach;
   (b) Anyone who fails to comply with these regulations may be charged with academic fraud.
3. Write comments and assumptions to get partial marks;
4. Write your answers in the space provided.
   (a) Use the back of pages if necessary;
   (b) You may not hand in additional pages.
5. Do not remove the staple holding the examination pages together;
6. Beware, poor hand writing can affect grades;

Marking scheme

Question	Maximum	Result
1	35	
2	15	
3	15	
Total	**65**	

Question 1 (35 marks)

A start-up company named **Gazouillis** wants to implement a breakthrough software system to exchange text and photographic messages. You have been tasked to create the prototype implementation that will be demonstrated to the investors.



The above UML diagrams give you an overview of this application. Follow the instructions below.

A. Implement the interface **Likeable**. It declares two methods: **like()** and **int getLikes()**. (5 marks)

B. Write the implementation of the abstract class **Post**. It implements the characteristics that are common to its sub-classes, here **TextPost** and **PhotoPost**. (10 marks)

- **Post** implements the interface **Likeable**.
- All the **Post** messages have a user name, a time stamp (of type **java.util.Date**), as well as a count for the number of likes.
- The value of the time stamp is automatically assigned when an object is created. Use **java.util.Calendar.getInstance().getTime()** to obtain a **Date** object representing the current time. A **Date** object has a method **toString()** that converts this date to a **String**.

```
Date rightNow = Calendar.getInstance().getTime();
System.out.println(rightNow);
```

- Each call to the method **like()** increases the number of likes for this message.

C. Implement the class **PhotoPost**. A **PhotoPost** is a specialized **Post**. It stores a file name and a caption. (5 marks)

D. Implement the class **TextPost**. A **TextPost** is a specialized **Post**. It stores a text message. (5 marks)

E. Write the implementation of the class **NewsFeed**. A **NewsFeed** object stores **Post** messages. (10 marks)

- It uses the dynamic array implementation seen in class to store **Post** messages.
- The constructor has two parameters, the initial capacity of the array and the capacity increment.
- Each time the array is full, the implementation should create a new array larger by the capacity increment.

- It has a method for adding a **Post** message. The message is added after the last message added.
- It has a method for returning the message found at a given index, **Post get(int index)**.
- It has a method **size** that returns the number of messages currently stored.

Implement all the necessary constructors. Each attribute must have a getter method. Here is a test program illustrating the use of these classes. You can assume that all the parameters are valid. Write your answers in the appropriate boxes.

```
public class Test {

    public static void main(String[] args) {

        NewsFeed messages;
        Post msg1, msg2;

        messages = new NewsFeed(100, 10);

        msg1 = new PhotoPost("David", "funny.png", "Birthday party");
        msg1.like();
        messages.add(msg1);

        msg2 = new TextPost("David", "Dinner at your place with Alexe");
        msg2.like();
        msg2.like();

        messages.add(msg2);
        messages.add(new TextPost("Nancy", "Okay"));

        for (int i=0; i<messages.size(); i++) {
            System.out.println(messages.get(i));
        }

    }
}
```

Here is the expected output.

```
PhotoPost: Sun Feb 24 11:31:08 2013, David, likes = 1, funny.png, Birthday party
TextPost: Sun Feb 24 11:31:08 2013, David, likes = 2, Dinner at your place with Alexe
TextPost: Sun Feb 24 11:31:08 2013, Nancy, likes = 0, Okay
```

Likeable

Post

TextPost

PhotoPost

NewsFeed

Question 2 (15 marks)

For the class **Q2**, you must implement the class method **boolean isReverse(Stack s1, Stack s2)**.

- The method **isReverse** returns **true** if and only if the stack designated by **s1** contains the same elements as that designated by **s2** and the order of the elements in the stack designated by **s1** is the reverse of the order of the elements in the stack designated by **s2**.
- Following a call to the method **isReverse**, the stack designated by **s1** must contain the same elements, in the same order, as it did before a call to **isReverse**.
- Following a call to the method **isReverse**, the stack designated by **s2** must contain the same elements, in the same order, as it did before a call to **isReverse**.
- You can assume that both **s1** and **s2** will not be **null**.
- An empty stack is considered the reverse of another empty stack.
- The parameters of the method **isReverse** are of type **Stack**, which is an interface.

For this question, there is an interface named **Stack**:

```
public interface Stack {
    public abstract void push(int item);
    public abstract int pop();
    public abstract boolean isEmpty();
}
```

- Notice that the parameter of the method **push** and the return value of the method **pop** are of type **int**.
- You cannot use arrays to store temporary data.
- Assume the existence of **DynamicStack**, which implements the interface **Stack**. It has one constructor and its signature is **DynamicStack()**.
- You must use objects of the class **DynamicStack()** to store temporary data.
- You do not know anything about the implementation of **DynamicStack**. In particular, you do not know if it uses an array or not.
- You can assume that **DynamicStack** can store an arbitrarily large number of elements.

Write your answer on the next page.

```
public class Q2 {
    public static boolean isReverse(Stack s1, Stack s2) {









    } // End of isReverse
} // End of Q2
```

(Question 2 continued)

Question 3 (15 marks)

A. True or false questions (5 marks)

(a) The Java program below displays the value **true**.
True or **False**.

```
public class Test {
    public static void main(String[] args) {
        int[] a, b;
        a = new int [100];
        a[0] = 5;
        b = a;
        b[0] = 6;
        System.out.println( a == b );
    }
}
```

(b) A local variable is confined to the method containing its declaration.
True or **False**.

(c) Referring to the class **Counter** below, the keyword **this** can be omitted without affecting the compilation or execution of the program.
True or **False**

```
public class Counter {
    private int value = 0;
    public Counter( int initialValue ) {
        this.value = initialValue;
    }
}
```

(d) A reference variable of type **T**, can reference an object of class **S**, if **S** is a superclass of **T**.
True or **False**

(e) A class can implement multiple interfaces.
True or **False**

B. Multiple choice questions (6 marks)

(a) _____ members of a base class are never accessible to a derived class.

i. public
ii. private
iii. protected
iv. a, b, and c
v. none of the above

Answer:

(b) _____ allows us to create new classes based on existing classes.

i. polymorphism
ii. inheritance
iii. method overloading
iv. the copy constructor
v. none of the above

Answer:

(c) Which of the following is the postfix representation of the infix expression:

$$((((9 + 7)/8) * 6) - 2)$$

i. 9 7 + 8 / 6 * 2 −
ii. 9 8 7 + / 6 * 2 −
iii. 9 7 8 6 2 + / * −
iv. 9 / 7 8 + 6 * 2 −

C. Following the guidelines presented in class, as well as the lecture notes, draw the memory diagrams for all the objects and all the local variable, and parameter of the method **Counts.main** following the execution of the statement "**cs = new Counts();**" (4 marks).

```
public class Word {

    private String word = null;
    private int count = 0;

    public Word(String word, int count) {
        this.word = word;
        count = count;
    }

}

public class Counts {

    private Word[] words;

    public Counts() {
        words = new Word[2];
        words[0] = new Word("ITI1121", 200);
        words[1] = new Word("ITI1521", 55);
    }

    public static void main(String[] args) {
        Counts cs;
        cs = new Counts();
        // here
    }

}
```

(blank space)