PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
FACULTY OF ARTS AND SCIENCE

AUGUST 2016 EXAMINATIONS

CSC 207H1Y
DURATION — 3 HOURS
NO AIDS ALLOWED

STUDENT NUMBER: |_|_|_|_|_|_|_|_|_|_|

LAST/FAMILY NAME: _____

FIRST/GIVEN NAME: _____

*Do* NOT *turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below.)

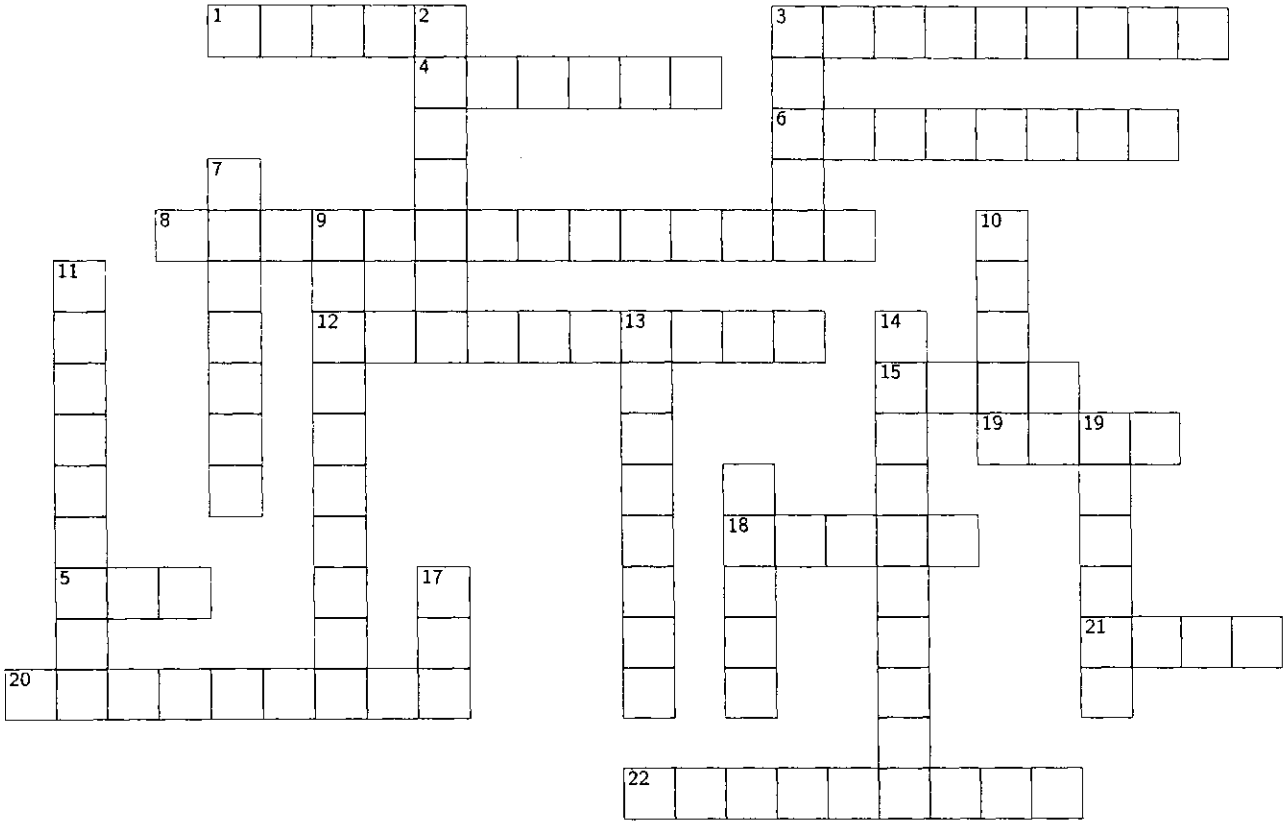
---

This test consists of 7 questions on 24 pages (including this one).
You must receive 40% on this examination to pass the course.
*When you receive the signal to start, please make sure that your copy of the test is complete.*

*Good Luck!*

## QUESTION 1. [10 MARKS]

This exercise deals with Observer pattern. Please solve the following crossword puzzle based on the clues shown below the puzzle. In order to get full marks you need to fill in 20 correct answers. Each correct answer earns 0.5 marks. The maximum number of marks you can earn from this exercise is 10 marks.



ACROSS 1 Subject is a _____ not an interface 3 ActionListener objects are _____ to the button 4 Implement this method to get notified 5 Observers can _____ themselves to Observables. 6 SalesOrder class from exercise 3 implements this interface 8 How to get yourself off the Observer list 12 You forgot this if you're not getting notified when you think you should be 15 One Subject likes to talk to _____ observers 18 Don't count on this for notification 19 The Design Patterns orginal book has been written by The _____ of Four 20 Observers are _____ on the Subject 21 Gang of _____ 22 A Subject is similar to a _____

DOWN 2 A JButton object is a _____ 3 You want to keep your coupling _____ 7 The Observer Design Pattern addresses the need to maintain consistency between _____ objects 9 _____ can manage your observers for you 10 Java framework with lots of Observers 11 Program to an _____ not an implementation 13 Observers like to be _____ when something new happens 14 The WeatherData class _____ the Subject interface 16 ActionListener objects handle _____ clicks 17 An observable may have methods to get and _____ its state 19 An object should be able to _____ other objects without making assumptions about who these objects are



CONT'D...

## QUESTION 2. [10 MARKS]

Answer the following questions:

### PART (A) [2 MARKS]

Consider the following code:

```
public class Stat1 {
   protected String name;
   public Stat1(String name) {
       this.name = name;
   }
   public static void doStaticWork() {
      System.out.println("I am a static
            method defined in Stat1");
   }
}
```

```
public class Stat2 extends Stat1{
   public Stat2(String name) {
      super(name);
   }
   @Override
   public static void doStaticWork() {
      System.out.println("I am a static
         method defined in Stat2");
   }
}
```

Q1: Will the code for both classes defined above compile? Answer "Yes" or "No": _____

Q2: If you answered "Yes", explain why. If you answered "No", write the line(s) of the code that cause the compilation error.

---

---

### PART (B) [2 MARKS]

In order to utilize the predefined Java classes HashMap and HashSet, what two methods inherited from class Object might need to be overridden?

---

---

PART (C) [3 MARKS]

Write next to each method call in main() the output that it prints:

```
class A {
   public void f(A a) { System.out.println("fa(A)"); }
   public void f(B b) { System.out.println("fa(B)"); }
}
class B extends A {
   public void f(A a) { System.out.println("fb(A)"); }
   public void f(B b) { System.out.println("fb(B)"); }
}
public class TypeTesting {
   public static void main(String[] args) {
      A a = new A();
      B b = new B();
      A ba=(A)b;
      // Write output next to each of the following:
      a.f(a);     // Your answer:
      a.f(b);     // Your answer:
      b.f(a);     // Your answer:
      b.f(b);     // Your answer:
      a.f(ba);    // Your answer:
      b.f(ba);    // Your answer:
      ba.f(a);    // Your answer:
      ba.f(b);    // Your answer:
      ba.f(ba);   // Your answer:
   }
}
```

PART (D) [1 MARK]

What is the most specific result of the following code:

```
Integer[] someInts = new Integer[100];
int sum = 0;
for ( Integer i : someInts )
{
   sum += i;
}
System.out.println( sum / someInts.length );
```

Check the box that corresponds to your answer. Do not guess. An incorrect answer will be punished by $-.5$ points. In clase you do not know the answer, please check the last box.

- ☐ 0
- ☐ The average of 100 integers
- ☐ NullPointerException
- ☐ The code will not compile
- ☐ I don't know

PART (E) [1 MARK]

A class which implements the `ActionListener` interface must implement which method? Check the box that corresponds to your answer. Do not guess. An incorrect answer will be punished by $-.5$ points. In clase you do not know the answer, please check the last box.

- ☐ `void handle( ActionEvent e )`
- ☐ `void actionPerformed( ActionEvent e )`
- ☐ `void eventDispatched( AWTEvent e )`
- ☐ `String getActionCommand( ActionEvent e )`
- ☐ I don't know

PART (F) [1 MARK]

Given the following method and class signatures:

```
public class A extends Exception {...}
public class B extends A {...}
public class C extends B {...}
public void doStuff() throws A,B,C
```

The following code does not compile. Why?

```
try {
   doStuff();
} catch(A a) {
   a.printStackTrace();
} catch(B b) {
   b.printStackTrace();
} catch(C c) {
   c.printStackTrace();
} finally {
   System.out.println("I love exceptions!");
}
```

Check the box that corresponds to your answer. Do not guess. An incorrect answer will be punished by −.5 points. In clase you do not know the answer, please check the last box.

☐ The catch blocks for exceptions of type B and C are unreachable.

☐ A finally block cannot be used with multiple catch blocks.

☐ B and C are not exception classes since they do not extend class Exception and therefore cannot be caught.

☐ The finally block fails to compile.

☐ I don't know

## QUESTION 3. [11 MARKS]

### PART (A) [3 MARKS]

Given the interface and class definitions from below, what are the methods that you definitely need to implement yourself in class MyClass?

```
interface I {
   public float mI(int a);
}
interface J extends I {
   public int mJ(int a);
   public Object mJJ(int a);
}
class C {
   public void mC(int a) {
      System.out.println(    hello  world    );
   }
}
class MyClass extends C implements J {
...
}
```

---

---

### PART (B) [1 MARK]

Given the following definitions, which assignments are legal?

```
class Box<T>{}
class SuperBox<T> extends Box<T>{}
(a). Box<Object> b = new Box<String>();
(b). Box<String> b = new SuperBox<String>();
(c). Box<Object> b = new SuperBox<String>();
```

Check the box that corresponds to your answer. Do not guess. An incorrect answer will be punished by −.5 points. In clase you do not know the answer, please check the last box.

☐ (a), (b), and (c).

☐ (b) only.

☐ (a) and (c) only.

☐ (a) only.

☐ I don't know

PART (C) [5 MARKS]

Assume we are given the following method:

```
public int B(String s) throws XException, YException {
   // some code ....
}
```

Write a public method called A that belongs to the same class as B. Method A should satisfy the following requirements:

- It takes a String parameter and returns an int
- It calls B, passing it the same string it was given
- If B throws a YExcpetion, A prints Yak!
- If B throws a XEception, A does'nt deal with it; A just passes it along
- If B executes without throwing any exception, A either:
  - Throws a ZEception if B's answer is less than zero, or
  - Simply returns B's answer if it is at least zero.

---

PART (D) [1 MARK]

What is the result of the following code within a single class where all relevant code has been shown?

```
private Date today;
public void someMethod( String name, String favColor )
{
   System.out.println( name + "   s favorite color on "
      + today.toString() + " is " + favColor );
}
//... Somewhere someMethod is called...
   String name = "Topato";
   String favColor = "Green";
   someMethod( favColor, name );
...
```

Check the box that corresponds to your answer. Do not guess. An incorrect answer will be punished by −.5 points. In clase you do not know the answer, please check the last box.

- ☐ This code will not compile
- ☐ Greens favorite color on Tue Dec 12 10:27:00 EST 2006 is Topato
- ☐ Topatos favorite color on Tue Dec 12 10:27:00 EST 2006 is Green
- ☐ NullPointerException
- ☐ I don't know

PART (E) [1 MARK]

Given a class Reindeer with the following signature:
`class Reindeer throws HoHoHoException{...}`
The following code throws an exception when it attempts to write 9 Reindeer objects to a file. Why?

```
File outFile = new File("Santa.txt");
FileOutputStream outFileStream = new FileOutputStream(outFile);
ObjectOutputStream outObjectStream = new ObjectOutputStream(outFileStream);
Reindeer r;
for(int i = 0; i < 9; i++){
   r = new Reindeer("Reindeer" + (i + 1));
   outObjectStream.writeObject(r);
}
```

Please do not guess. Incorrect answers will be punished with $-.5$ points. In case you do not know the answer, please check the "I don't know" box.

☐ The Reindeer class does not implement the Serializable interface

☐ Only 8 Reindeer objects are written

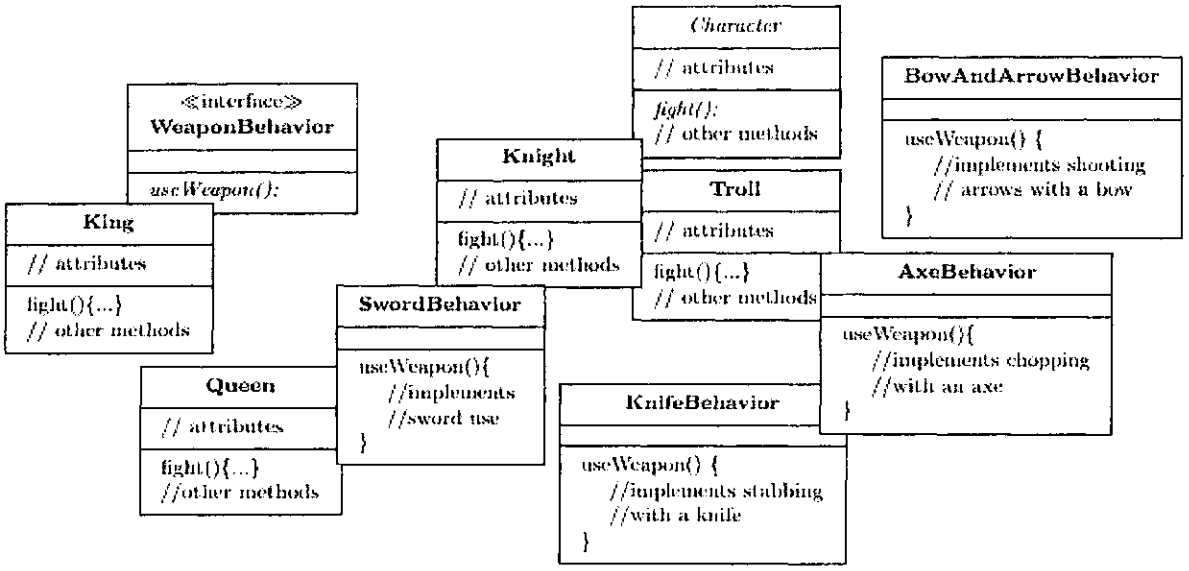
☐ The file Santa.txt is not a data file and we can only write objects to data files

☐ We should use a DataOutputStream instead of an ObjectOutputStream

☐ I don't know

## QUESTION 4. [10 MARKS]

Below you sill find a mess of classes and interfaces for an action adventure game. You'll find classes for game charaters along with classes for weapon behaviors the characters can use in the game. Each character can make use of one weapon at the time, but can change weapons at any time during then game. You may assume all variables and methods are public, so no need to provide the UML code for the visibility modifiers.




Please complete the following tasks:

### PART (A) [1 MARK]

Identify the correct design pattern we should use here.

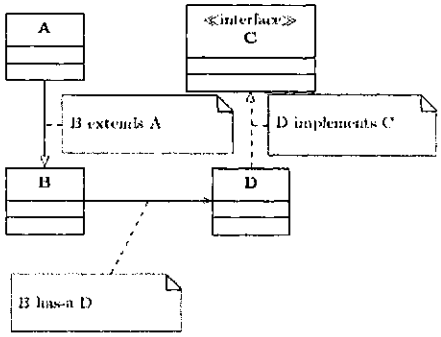
---

---

### PART (B) [5 MARKS]

In the provided space complete the following tasks:

- Arrange the classes.
- Identify one abstract class, one interface, and eight classes.
- Draw the correct arrows between classes:




PART (C) [4 MARKS]

Add a variable of type `WeaponBehavior` named `weapon` in the correct class(es). Write the code for the method `setWeapon(WeaponBehavior w)` and put it in the correct class(es).

## QUESTION 5. [10 MARKS]

### PART (A) [2 MARKS]

Using regular expressions, write the code for the following function which returns `true` if the argument is a prime number, otherwise returns `false`. No credit will be given for solutions that do not use regular expressions.

---

```
public static boolean prime(int n) {
  // Your code goes here
```

---

HINT: By definition, a prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. That means if $n = k * m$ for natural numbers $k, m \notin \{1, n\}$ then $n$ is not a prime.

### PART (B) [2 MARKS]

Using regular expressions write the code for the following method, intended to extract all integers from a string and return them in the form of an `ArrayList`. No credit will be given for solutions that do not use regular expressions.

---

```
public ArrayList<Integer> extract(String str) {
// your code goes here
```

PART (C) [2 MARKS]

Complete the Agile Manifesto: We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

PART (D) [2 MARKS]

Recall the GaussInteger class from Exercise 1:

```
public class GaussInteger {
        private int real, imag;

    public GaussInteger(int real, int imag) {
        this.real = real;
        this.imag = imag;
        }
        public int getImag() {
                return imag;
        }

// more code
}
```

Write a reflection unit test for `getImag()` method.

PART (E) [2 MARKS]

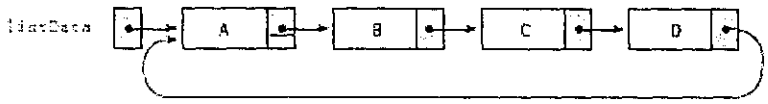
This question deals with unit testing of the `toString()` method for the `GaussInteger()` class described above. Write a unit test that tests the `toString()` gives a representation of the imaginary unit that makes sure that a correct implementation of the `toString()` method would represent the imaginary unit as `"i"`. The imaginary unit object can be generated by this statement:

```
GaussInteger imagUnit = new GaussInteger(0, 1);
```

---

## QUESTION 6. [9 MARKS]

In this question you will implement an unbounded circular list. By definition, a *circular linked list* is a list in which every node has a successor; the *last* element is succeeded by the *first* element:




Our circular list is allowed to hold objects of the same type. You will accomplish this using Java generics. In order to construct our list, we will make use of a helper class `Node<T>` defined as follows:

```
public class Node<T>
{
  private Node<T> link;
  private T info;
  public Node(T info)
  {
    this.info = info;
    link = null;
  }
  // In addition, we will assume we have defined the getters and setters the usual way.
 }
```

Here is an example of use of the class `CircularList` that you will write.

```
public class UseCircular {
   public static void main(String[] args) {
      CircularList<String> myList = new CircularList<>();
      myList.add("A");
      myList.add("B");
      myList.add("C");
      myList.add("D");
      System.out.println("Below is myList formatted using toString():");
      System.out.println(myList);
   }
}
```

This program should product the following output:

```
Below is myList formatted using toString():
List: [A, B, C, D]
```

Complete the implementation of `CircularList`. Do not forget that this class should be generic. Do not use any of Java Collection classes.

```
// Complete the class declaration.
```

---

---

```
// Add instance variables, if needed
```

---

```
/**
 * Constructs an empty circular list
 */
```

---

```
/**
 * Adds an element to this list.
 * @param element - to be added to the list
 */
```

```
/**
* The size() method returns the number of the elements on the list
* @return an integer
*/
```

---

## QUESTION 7. [10 MARKS]

This question uses class `CircularList` fromthe previous question. You will now use the iterator design pattern to add iterator support for `CircularList`. Here is an example use of a for loop to iterate over the elements of a `CircularList`.

```
        CircularList<String> myList = new CircularList<>();
        myList.add("A");
        myList.add("B");
        myList.add("C");
        myList.add("D");
        System.out.println("Below is my list processed using a for loop");
        for(String s : myList) System.out.print(s);
//
// Output from this code is:
Below is my list processed using a for loop
ABCD
```

### PART (A) [1 MARK]

Show how you need to modify the declaration of `CircularList`.

---

---

### PART (B) [9 MARKS]

Now add whatever is necessary to the class `CircularList` to ensure that the foreach loop above works. You may continue on the next page.

---

Youmay use this page to write the solution to the question 7..

This page is left for scratch work.



CONT'D...

# 1: _____/10

# 2: _____/10

# 3: _____/11

# 4: _____/10

# 5: _____/10

# 6: _____/ 9

# 7: _____/10

TOTAL: _____/70