Computing Science 114 Midterm Exam #1

October 11, 2000

Section: A1 - M. Szafron

Last Name: _____

First Name: _____Student ID#:_____

Instructions:

- Write your id number on every page where indicated.
- You have **50 minutes** to complete this exam
- No books, notes, calculators or other aids are allowed in this exam except as provided
- Please remove all hats, sunglasses, caps, and anything else that obscures your face
- Please take appropriate measures to shield your answers from others around you
- Once you leave the examination room, you will not be re-admitted
- The exam has been printed single-sided so you can use the back of each page for your rough notes and calculations, as well there is an extra blank page provided at the end (page 10)
- All answers are to be done on this paper only
- There should be 10 pages in your exam paper (including this cover page) and an Adventure code handout.

E 05157
CMPUT 114 (A1)
SZAFRON, M.
OCT 00 MIDTERM 1
PAGES: 10

Results:

Question 1 (Multiple choice)	**/ 15**
Question 2 (Fill in the blanks)	**/ 10**
Question 3 (Tracing)	**/ 26**
Question 4 (Write the code)	**/ 29**
Question 5 (Adventure Code version 3)	**/ 20**
Total:	**/100**

Please note: after the exams have been graded, 50% of the papers will be randomly selected, photocopied & retained

#1 [15 marks – 3 each] Please circle EXACTLY ONE choice as the best answer to each question:

1.1) Which ONE of the following kinds of variables has the largest scope?

a. *public static
b. private static
c. private instance
d. local
e. method parameter

1.2) Which ONE of the following variables has the shortest lifetime?

a. public static
b. private static
c. private instance
d. *local
e. *method parameter

1.3) Which ONE of the following is an example of a run-time error?

a. omitting a curly bracket at the end of method body
b. omitting the return type of a method
c. printing "hello" when you wanted to print "goodbye"
d. *giving different names to the Java program file and the Java target file
e. forgetting a semi-colon at the end of a statement

1.4) Here are the BNF rules for a segment:

 ::= <front><inside><back>
<front> ::= 1 | 2 | 3
<inside> ::= <hard>|<easy>
<hard> ::= <easy><easy>
<easy> ::= 0
<back> ::= 8 | 9

Which of the following is not a legal segment?

a. 109
b. 2009
c. *303
d. 1008
e. *10008

1.5) Which ONE of the following is false about a Java primitive value?

a. it can have a literal value reference to it
b. it can be returned as result of a message
c. it can be an argument to a message
d. *it can be the receiver of a message
e. it can be used in a comparison

#2. [10 marks – 2 each] Fill in the blanks with a term that best finishes the statement.

1. Methods of the same name can be distinguished from each other by their parameter list.
2. A boolean value is an example of a(n) primitive type.
3. The keywords private or public indicate the visibility or scope of an instance method or instance variable.
4. An object whose state can be changed is called mutable
5. Method dispatch is the association of messages to instance methods.

#3 [26 marks] Consider the following Java Class:

```
public class Trace {

/* Instance Variables */
     private int a;
     private int b;
     private int c;

/* Constructors */
     public Trace(int v1, int v2, int v3) {
           this.a = v1;
           this.b = v2;
           this.c = v3;
     }
/* Instance Methods */
     public void m1(int a) {

           this.b = a + this.c;
           this.c = a + this.b;

           System.out.print(a);
           System.out.print("  ");
           System.out.print(this.b);
           System.out.print("  ");
           System.out.println(this.c);
     }
     public void m2(int c) {

           this.a = this.b + c;
           System.out.print(this.a);
           System.out.print("  ");
           System.out.println(c);
           this.m1(c);
     }

     public void m3(int a) {

           int d;

           d = a + this.b + this.c;
           System.out.print(a);
           System.out.print("  ");
           System.out.print(this.b);
           System.out.print("  ");
           System.out.print(this.c);
           System.out.print("  ");
```

```
            System.out.print(d);
            System.out.print("  ");
            System.out.println(this.a);
        }
    }
```

a) What is the output of the program segment:

```
Trace      trace1;
trace1 = new Trace(2, 4, 6);
trace1.m1(8);
```

OUTPUT

```
8   14   22
```

b) What is the output of the program segment:

```
Trace      trace2;
trace2 = new Trace(1, 3, 5);
trace2.m2(10);
```

```
13   10
10   15   25
```

c) What is the output of the program segment:

```
Trace      trace3;
trace3 = new Trace(2, 4, 6);
trace3.m3(12);
```

```
12   4   6   22   2
```

#4. [29 marks] Consider a class called Athlete. Every Athlete has a name, an event, a list of medals and the total number of medals that he/she has won. When the Athlete is created, he/she is given a name and an event. At the same time, the list of medals and the number of medals won are initialized. There are various record keeping and reporting activities carried out by the methods in this class. The exact description of what each method does is given in the /*comments*/ of each method. There is a second class called OlympicGames. This class is the main program and it creates 2 Athletes, gives medals to them and reports on their medal standings. The output of this main program appears below.

```
Welcome to the Year 2000 Olympic Summer Games

Fred's Medal List:
[Silver]

Barney's Medal List:
[Gold, Bronze]

Name       Event       Last Medal Won    Total Number of Medals
Fred       Shotput        Silver               1
Barney     Javelin        Bronze               2
```

Here is the code for the class Athlete that creates an Athlete and displays it. **Fill in the blanks with the missing code.**

```
import java.util.*;
public class Athlete {
/* Each of my instances represents an Athlete that has a name, an event, a list of
medals won and the number of medals won. */

/* Private Instance Variables */
    private String name;
    private String event;
    private Stack medals;
    private int medalCount;

/* Constructor */
    (1 mark)
    public Athlete (String athleteName,String OlympicEvent){
    _____){
    /*  Initialize me to have the given name and event. Also initialize me to have no
        medals and no medal count.
    */
    (2 marks)
        this.name = athleteName;
        this.event = olympicEvent;
        this.medals = new Stack();
        this.medalCount = 0;}

    /* Public Instance Methods */

    public void addMedal(String newMedal) {
```

```
/*
      Add the given medal to my Stack of medals and increment how many I have.
*/
(2 marks)
      this.medals.push(newMedal);
      this.medalCount = this.medalCount + 1;
}
public int medalTotal() {
/*   Return the number of medals in me. */

(1 mark)
      return this.medalCount;
 }

public String name() {
/*  Return my name.  */

(1 mark)

      return this.name;

public String sport(){
/*
      Return my event.
 */
(1 mark)
      return this.event;
}

public void displayLastMedalWon() {
/*
      Display on the screen the last medal I won.
*/
(2 marks)

      System.out.print(this.medals.peek());

}

public void medalList(){
/*
      Diplay all the medals I have won.
*/
(1 marks)

      System.out.println(this.medals);
}
```

```
}
*****************************************************************************************

public class OlympicGames {
/*
     Tests the class Athlete.
*/
     public static void main(String args[]) {
     /*
          Create 2 Athletes and displays reports about those Athletes.
     */
     (1 marks)

          Athlete     athlete1;

          Athlete     athlete2;
          int  totalMedals;

     (3 marks)
          athlete1 = new Athlete("Fred", "Shotput");
          athlete1.addMedal("Silver");

          athlete2 = new Athlete("Barney, "Javelin");
          athlete2.addMedal("Gold");
          athlete2.addMedal("Bronze");

          System.out.println("Welcome to the year 2000 Olympic Summer Games);
          System.out.println();  //prints a blank line

Note:  The remainder of the code is worth 1 mark per blank line.

          System.out.print(athlete1.name());
          System.out.println("'s Medal List:");

          Athlete1.medallist();
          System.out.println();

          System.out.print(athlete2.name());

          System.out.println("'s Medal List:");

          Athlete2.medallist();
          System.out.println();

          System.out.print("Name      ");
          System.out.print("Event     ");
          System.out.print("Last Medal Won");
```

```
            System.out.println("Total Number of Medals");
            System.out.print(athlete1.name());
            System.out.print("     ");
            System.out.print(athlete1.sport());
            System.out.print("     ");

            athlete1.displayLastMedalWon();
            System.out.print("     ");

            totalMedals = athelete1.medalTotal();

            System.out.println(totalMedals);

            System.out.print(athlete2.name());
            System.out.print("     ");
            System.out.print(athlete2.sport());
            System.out.print("     ");

            athlete2.displayLastMedalWon();
            System.out.print("     ");

            totalMedals = athlete2.medalTotal();

            System.out.println(totalMedals);
        }
    }
```

#5. [20 marks – 2 each] Adventure Code. Consider the code listing of the Adventure program Version 3 (you should have been given a copy of this code with your exam paper. If you do not have a copy, please ask a proctor to bring you one). When answering the following questions, be careful to distinguish between the Adventure class and the Adventurer class.

a) In the `play()` method of the Adventure class, what is the declared type of the object that "`this`" is bound to?

Answer = Adventure

b) In the `name()` method of the Adventurer class, what is the declared type of the object that "`this`" is bound to?

Answer = Adventurer

Consider this statement from the `enterRoom()` method of the Adventure class:

```
adventurer.gainTokens(myTokens.intValue());
```

c) How many message expressions are there in this statement?

Answer = 2

d) What is the declared type of `myTokens`?

Answer = Integer

e) What is the type of the parameter used by the `gainTokens` method in the Adventurer class?

Answer = int

f) What is the return type of the `enterRoom()` method in the Adventure class?

Answer = void

g) Which method contains the code which **invokes** the Adventurer constructor?

Answer = greeting (in Adventure Class)

Consider the following statement in the `enterRoom()` method of the Adventure class.

```
myTokens = Keyboard.in.readInteger();
```

h) What is the exporting class of the public static variable `in`?

Answer = Keyboard

i) Which class contains the code for the `readInteger()` method?

Answer = Keyboard

j) In the Adventure class, in the method `greeting()`, what is the scope of the variable `playerName`?

Answer = local

A blank page for your use.