Surname: ________________________________

Given Name: ____________________________

ID Number: _____________________________

User Name: _____________________________

University of Waterloo
CS 133 Fall 2007

Midterm Exam

Monday, October 22, 7:00 p.m. - 9:00 p.m. **Time: 2 Hours**
Instructors: Leila Chinaei, Caroline Kierstead, Brad Lushman **Permitted Aids: None**

Signature:
____

Submit this booklet at the end the examination.

Instructions: (Read Carefully)

1. You should have this Exam Booklet and a Reference Sheet.
2. Please ensure that the information recorded on the exam label is correct.
3. Cheating is an academic offense. Your signature on this page indicates that you understand and agree to the University's policies regarding cheating on exams.
4. Questions will **not** be interpreted. Proctors will confirm or deny question errors only. If you consider the wording of a question to be ambiguous or lacking critical information, read it again carefully, looking for clues to clarify the intent of the question. If you still are unsure, clearly **state *reasonable* assumptions** (which do not trivialize the question) and proceed to answer the question to the best of your ability.
5. Marks will **not** be given for documentation. If you have extra time and feel that it would improve the readability of your code, then include it.
6. Use the mark allocations to judge the amount of time you should spend on a question as well as the quantity of information you present in your solution.
7. Make sure that your code follows the principles of good programming and program development as taught in the course.

Question	Mark	Value	Marker
1		**7**	
2		**10**	
3		**18**	
4		**9**	
5		**10**	
6		**3**	
7		**6**	
8		**6**	
9		**8**	
10		**9**	
11		**13**	
Total		**99**	
Verified			

Name (printed): ____________________________ ID Number: _______________

Question 1: Public and Private (7 marks)

a) There are three reasons why attributes in a class should be declared private instead of public. Explain two of the three reasons.

Simplicity: it's clear how values are updated and retrieved. More to the point, it simply reduces the amount of information you have to manage while you write programs. We don't care how a method works so long as it does. You don't know how nextLine() in Scanner works, yet you can use it to read text from the keyboard (much like most people don't know how their car works yet they can drive it).

Security: you can't see it, you can't change it (directly).

Stability: by controlling access to the instance variables, we can isolate users from the specific implementation of a class. We can replace a method with another version that behaves in the same way, and this change won't affect users because none of their code depends on the implementation details. Many Java methods have changed since version 1, yet a good deal of the code written with early versions still works.

b) How should a helper method be declared in a class (public/private)? private

c) Can a helper method be called by a private method within the same class (yes/no)? yes

d) Can a helper method be called by more than one method within the same class (yes/no)? yes

e) Can a helper method be called by more than one public method within the same class (yes/no)? yes

f) Can a helper method be called by a method within another class (yes/no)? no

Name (printed): ___________________________ ID Number: _______________

Question 2: String Manipulation (10 marks)

a) **[4 marks]** Write a method called **`reverseString`** that takes a string as a parameter and returns a new string which is equal to the original string with the characters in reverse order. For instance the reverse of "**`Hello`**" would be "**`olleH`**". Restrict yourself to String methods and operations only. Complete the method signature and the body of the method below:

```
public String reverseString( String forward ) {

   String backward = "";
   for (int i=0; i < forward.length(); i++) {
      backward = forward.charAt(i) + backward;
   }
   return backward;
}
```

b) **[6 marks]** One simple way to encrypt text is to write all of the even-index (starting at 0) characters, followed by all of the odd index characters, ignoring punctuation and whitespace. For example, we might encode "**`I love CS133`**" as "**`IoeS3lvC13`**". Write a method called encrypt that takes a String as a parameter and returns a new String object that represents the result of encrypting the parameter String. Complete the method signature and the body of the method below::

```
public String encrypt( String source ) {

   String odds = "";
   String evens = "";
   boolean isEvens = true;

   for (int i=0; i < source.length(); i++) {
      char c = source.charAt(i);
      if (Character.isLetter(c) || Character.isDigit(c)) {
         if (isEvens) {
            evens += c;
         } else {
            odds += c;
         }

         isEvens = !isEvens;
      }
   }

   return evens + odds;
}
```

Name (printed): ________________________ ID Number: ______________

Question 3: Tracing (18 marks)

a) **[4 marks]** For each part, **either** check the box that indicates that a compiler error would occur, **or** write the output on the line associated with the code fragment.

```
public class X {
   private void foo() {
      System.out.print( "A" );
   }
   public void bar() {
      System.out.print( "B" );
      foo();
   }
   public static void main( String [] args ) {
      X x = new X();
      System.out.print( "C" );
      x.bar();
   }
}
```

Compiler Error: [ ]
Output:

`CBA`

```
public class X {
   private void foo() {
      System.out.print( "A" );
   }
   public void bar() {
      System.out.print( "B" );
      foo();
   }
}
public class Y {
   public void banana() {
      X x = new X();
      System.out.print( "C" );
      x.foo();
   }
   public static void main( String [] args ) {
      Y y = new Y();
      System.out.print( "C" );
      y.banana();
   }
}
```

Compiler Error: [✓]
Output:

```
public class X {
   private int orange = 0;
   public void bar() {
      orange++;
      System.out.print( "B" + orange );
   }
   public static void main( String [] args ) {
      X x = new X();
      System.out.print( "C" );
      x.bar();
   }
}
```

Compiler Error: [ ]
Output: `CB1`

```
public class X {
   public int orange = 0;
   public void bar() {
      orange++;
      System.out.print( "B" );
   }
}
public class Y {
   public void banana() {
      X x = new X();
      System.out.print( "C" );
      x.orange;
   }
   public static void main( String [] args ) {
      Y y = new Y();
      System.out.print( "D" );
      y.banana();
   }
}
```

Compiler Error: [✓]
Output:

Name (printed): ______________________ ID Number: ____________

b) [14 marks] What do the following code fragments print? In some cases, spaces and empty lines are important in the final answer and must be clearly shown. To make this clear, you can indicate the presence of a space using the underscore character ('_') and indicate an empty line with the text "<Blank line>". If we have any doubt, you will lose marks. If the code generates an error during execution, indicate the nature of the error. However, you must still note any output the program makes before the error occurs. Note, the symbol: ↵ refers to the newline character when it appears in the input.

```
public class X {
  public static void main( String [] args ) {
    System.out.println("Hi + \"CS133\" + Students");
  }
}
```

Runtime Error: ☐
Output:

```
Hi + "CS133" + Students
```

```
public class X {
   public static void main( String [] args ) {
      System.out.println("Hi ");
      System.out.print("CS133");
      System.out.println("Students");
   }
}
```

Runtime Error: ☐
Output:

```
Hi_
CS133Students
```

```
public class X {
   public static void main( String [] args ) {
      String s="Hi CS133 Students!";
      String t=null;

      System.out.println(s.substring(3,17));
      s=t;
      System.out.println(s.charAt(1));
   }
}
```

Runtime Error: [✓]
Output:

```
CS133 Students
```

```
public class Rectangle {
   public static int area(int height, int width) {
     return height * width;
   }
   public static void main( String [] args ) {
      System.out.println("The result of area is:  " +
                                Rectangle.area(4, 5) );
   }
}
```

Runtime Error: ☐
Output:

```
The result of area is: 20
```

Name (printed): ______________________ ID Number: ____________

```
public class X {
  public static void main( String [] args ) {
    int mark=85;

    if(mark > 90) {
      System.out.println("Excellent");
    }else if(mark < 90 || mark > 80) {
      System.out.println("Good job!");
    }else if (mark == 85 ) {
      System.out.println("Good job! your mark is
              exactly 10 marks above the average");
    }else if (mark > 70) {
      System.out.println("Doing okay.");
    }else if(mark > 50) {
      System.out.println("Passed");
    } else {
      System.out.println("Failed");
    }
  }
}
```

Runtime Error:

Output:

```
Good job!
```

```
public class X {
   public static void main( String [] args ) {
      for (int i=10; i >0 ; i=i/2) {
         System.out.print(i + “,”);
      }
     System.out.println(“Good job!”);
   }
}
```

Runtime Error:

Output:

```
10,5,2,1,Good job!
```

```
import java.util.*;
public class X {
   public static void main( String [] args ) {
      String name;
      int first;
      String second;

      Scanner keyBoardInput=new Scanner(System.in);
      System.out.println("Enter your name:");
      name=keyBoardInput.nextLine();

      System.out.println("Enter two numbers on different lines:");
      first=keyBoardInput.nextInt();
      second=keyBoardInput.nextLine();

      System.out.println("Hi:" + name);
      System.out.println("first + second = " + first + second);
   }
}
```

Input:

```
Anne John ↵
13 ↵
31 ↵
```

Runtime Error:

Output:

```
Enter your name:
Enter two numbers on different lines:
Hi:Anne John
first + second = 13
```

Name (printed): ______________________ ID Number: ____________

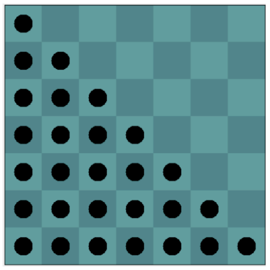
Question 4: Looping (9 marks)

a) **[4 marks]** Complete the following method which reads in an integer n and prints out the value of n factorial ($n \times (n-1) \times \ldots \times 1$). Assume the appropriate import statements already exist.

```
public void printFactorial() {

   Scanner input = new Scanner( System.in );
   int n = input.nextInt();
   int result = 1;
   while ( n > 0 ) {
      result = result * n;
      n--
   }
   return result;
}
```

b) **[5 marks]** Write a method to create and display the following pattern on a 7x7 board:



```
public void drawBoard() {

   Board b = new Board( 7, 7 );
   for ( int r = 0; r < 7; r++ ) {
      for ( int c = 0; c < 7; c++ ) {
            if ( r >= c ) {
               b.putPeg( Board.BLACK, r, c );
            }
      }
   }
}
```

Question 5: Java Classes (10 marks)

Fill in the blanks in the following class definition:

```
public class AClass {

  [ private      ] int member;  // Instance variable

  // Mutator method
  public[    void    ] setMember ( [int         ] member) {
      [    this.member          ] = member;
  }


  // Accessor method
  public[ int        ] getMember ( [             ] ) {
     return member;
  }

  // Constructor -- sets member field to the given value
  public[             AClass ]   ( [int value      ] ) {
     [    member  ] = [   value    ] ;
  }
}
```

Question 6: References and the Memory Model (3 marks)

Consider the following class definitions:

```
public class C {
  // Some instance variables and methods
}

public class D {
  C c;

  // Some constructor and method definitions.

  public C getC () {
    return c;
  }
}
```

Explain briefly why the method **getC** might be dangerous?

The value returned is a reference to the actual object c within the given instance of D, not a copy. Hence the user can call mutator methods on the object c, thereby modifying the D-instance's internal state without calling D's methods.

Name (printed): ____________________________ ID Number: ________________

Question 7: References and the Memory Model (6 marks)

Consider the following code:

```
public class C {
  public int x;
}

public class D {
  public static void f (C c, int y) {
    System.out.println(c.x);
    c.x = y;

    y++;
    System.out.println(c.x);
    c = new C();
    c.x = y+2;

    System.out.println(c.x);
  }

  public static void main (String[] args) {
    int z = 4;
    C c = new C();
    c.x = 3;

    System.out.println(c.x);
    f(c, z);

    System.out.println(c.x);
    System.out.println(z);
  }
}
```

What would be printed to **System.out** if this program were run? In some cases, spaces and empty lines are important in the final answer and must be clearly shown. To make this clear, you can indicate the presence of a space using the underscore character ('_') and indicate an empty line with the text "<Blank line>".

```
3
3
4
7
4
4
```

Name (printed): ______________________ ID Number: ____________

Question 8: References and the Memory Model (6 marks)

Consider the following classes. Implement the **.equals** method inside class C, such that **c1.equals(c2)** if and only if **c1.b.a.n == c2.b.a.n**. You may add public methods to any of these classes, but do not provide any additional accessor methods.

```
class A {
   private int n;

   public boolean equals(A other) {
      return this.n == other.n;
   }

}

class B {
   private A a = new A();

   public boolean equals(B other) {
      return this.a.equals(other.a);
   }


}

class C {
   private B b = new B();

   public boolean equals(C other) {
      return this.b.equals(other.b);
   }

}
```

Question 9: Overloading and Overriding (8 marks)

The class **`Rectangle`** contains the instance variables **`height`**, **`width`**, and **`colour`**. It implements the following methods:

1. Rectangle();
2. Rectangle(int size) ; // Creates a square where height = width = size
3. Rectangle(int height, int width);
4. Rectangle(int height, int width, Colour colour);
5. Rectangle(Rectangle otherRectangle);
6. void set(Colour colour);
7. void set(int size);
8. void set(int height, int width) ;
9. void set(int height, int width, Colour colour) ;
10. void draw(Board board);
11. String toString();

a) [2 marks] Give the line number (or numbers) of all the methods that demonstrate overloading.
1-5, 6-9

b) [2 marks] Give the line number (or numbers) of all the methods that demonstrate overriding.
1, 11

c) [2 marks] Is it possible to have **`Rectangle`** implement the following pair of methods? Explain why or why not.

```
int get();
Colour get();
```

No, since Java cannot distinguish methods solely based on return-type, so the number and types of parameters must differ.

d) [2 marks] Is it possible to have **`Rectangle`** implement the following pair of methods? Explain why or why not.

```
int get( int whichField );
Colour get();
```

Yes, since the methods differ in terms of the numbers and types of their formal parameters.

Name (printed): ______________________ ID Number: ____________

Question 10: Multiple Constructors (9 marks)

Assume the **`Rectangle`** class from question 9 has instance variables **`height`**, **`width`** and **`colour`**, and that it has default height, width and colour constants (**`DEFAULT_HEIGHT`**, **`DEFAULT_WIDTH`**, and **`DEFAULT_COLOUR`**, respectively). If one or more of the values is not supplied in the constructor, we have a reason to introduce a helper method to help initialize the object. Write the helper method, and show how it would be used in the constructors.

```
private void helpSet( int height, int width, Color colour ) {
     this.height = height;
     this.width  = width;
     this.colour = colour;

}

Rectangle() {
     helpSet( DEFAULT_HEIGHT, DEFAULT_WIDTH, DEFAULT_COLOUR );

}

// Creates a square where height=width=size
Rectangle( int size ) {
     helpSet(size, size, DEFAULT_COLOUR );

}

Rectangle(int height, int width, Color colour ) {
     helpSet( height, width, colour );

}

Rectangle( Rectangle otherRectangle) {
     helpSet( otherRectangle.height, otherRectangle.width,
              otherRectangle.colour );

}
```

Write the **`set( int size )`** method so it uses the helper method you wrote as well.

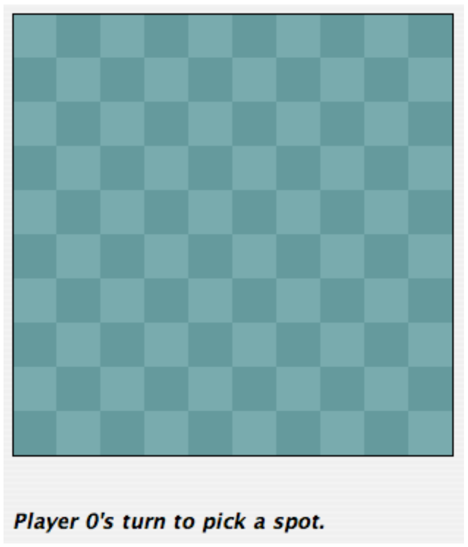
```
public void set( int size ) {
     helpSet( size, size, this.colour );
}
```

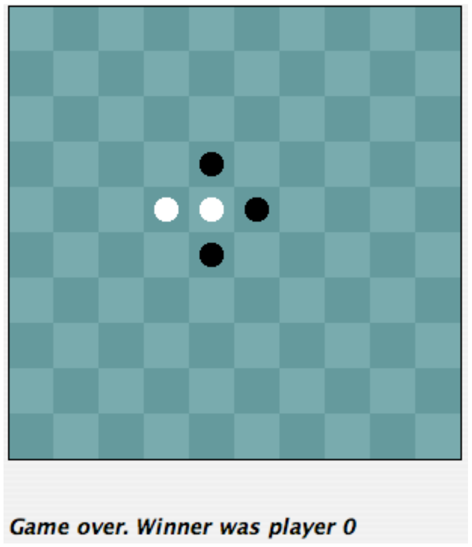
Name (printed): ___________________________ ID Number: _______________

Question 11: Top-down Design (13 marks)

"SOS" is a two-player, pencil-and-paper game. The players draw a grid, and then take turns placing a letter on the board. The first player can only write the letter 'S', and the second player can only write the letter 'O'. The first person who completes the word "SOS" in any direction on the board wins the game.

This game can be easily simulated using the **`BoardPlusModel`** class (see the reference sheet) and a 10x10 board if we have the player placing a black peg (**`Color.BLACK`**) instead of writing 'S', and the other player a white peg (**`Color.WHITE`**) instead of writing 'O'. The hard part of writing the game is checking for the presence of a black peg, followed by a white peg, followed by a black peg **in any possible direction** on the board.


Player 0's turn to pick a spot.


Game over. Winner was player 0

For this question, you will implement the method **`isGameOver`**, which takes no parameters and returns true when an "SOS" pattern appears somewhere on the board. Your solution **must** show evidence of top-down design for full marks. Note that efficiency will also matter—your code must stop searching the board as soon as the first valid pattern is found. Your search must take into account where the peg is in relation to the board edges. You should not search in that direction if it is not possible to find the peg pattern in the amount of space on the board. Keep your search as simple as possible to avoid introducing potential errors. You have two instance variables to work with, the constant **`BOARD_SIZE`**, and the instance of **`BoardPlusModel`**, **`gameBoard`**.

Complete the code on the following page. Assume the following method has already been written and implemented for you (so you may use it without writing it):

```
/* pre:  gameBoard != null and gameBoard.getColour(…) != null for each (x,y) coordinate pair:
 *             (x1,y1), (x2,y2) and (x3,y3)
 * post: returns true iff (x1,y1)==Color.BLACK && (x2,y2)==Color.WHITE && (x3,y3)==Color.BLACK
 */
private boolean isSOS(int x1, int y1, int x2, int y2, int x3, int y3);
```

```
// Assume this method is already written
private boolean isSOS(int x1, int y1, int x2, int y2, int x3, int y3);

private boolean checkHorizontal( int row, int col ) {
   if ( row < 0 || row >= BOARD_SIZE || col < 0 || col >= BOARD_SIZE ) {
      return false;
   } // if
   if ( BOARD_SIZE - col < 3 ) {
      return false;
   } // if
   return isSOS( row, col, row, col+1, row, col+2 );
} // checkHorizontal

private boolean checkVertical( int row, int col ) {
   if ( row < 0 || row >= BOARD_SIZE || col < 0 || col >= BOARD_SIZE ) {
      return false;
   } // if
   if ( BOARD_SIZE - row < 3 ) {
      return false;
   } // if
   return isSOS( row, col, row+1, col, row+2, col );
} // checkVertical

private boolean checkDiagonalTopLeftToBottomRight( int row, int col ) {
   if ( row < 0 || row >= BOARD_SIZE || col < 0 || col >= BOARD_SIZE ) {
      return false;
   } // if
   return isSOS( row, col, row+1, col+1, row+2, col+2 );
} // checkDiagonalTopLeftToBottomRight

private boolean checkDiagonalBottomLeftToTopRight( int row, int col ) {
   if ( row < 0 || row >= BOARD_SIZE || col < 0 || col >= BOARD_SIZE ) {
      return false;
   } // if
   return isSOS( row, col, row-1, col+1, row-2, col+2 );
} // checkDiagonalBottomLeftToTopRight
```

```
private boolean checkPosition( int row, int col ) {
   boolean gameOver = false;
   gameOver = gameOver || this.checkHorizontal( row, col );
   if ( !gameOver ) {
      gameOver = gameOver || this.checkVertical( row, col );
   } // if
   if ( !gameOver ) {
      gameOver = gameOver ||
         this.checkDiagonalTopLeftToBottomRight( row, col );
   } // if
   if ( !gameOver ) {
      gameOver = gameOver ||
         this.checkDiagonalBottomLeftToTopRight( row, col );
   } // if
   return gameOver;
} // checkPosition

private boolean isGameOver() {
   boolean gameOver = false;
   for ( int i = 0; !gameOver && i < BOARD_SIZE; i++ ) {
      for ( int j = 0; !gameOver && j < BOARD_SIZE; j++ ) {
         if ( gameBoard.getColour( i, j ) != null
               && gameBoard.getColour( i, j ) == Board.BLACK ) {
            gameOver = gameOver || this.checkPosition( i, j );
         } // if
      } // for
   } // for
   return gameOver;
} // isGameOver
```