CS 180 Fall 2006 Final Exam

There are 30 multiple choice questions. Each one is worth 3 points. There are 5 programming questions worth a total of 110 points.

Answer the multiple choice questions on the bubble sheet given and the programming questions on the exam booklet.

Fill in the Instructor, Course, Signature, Test, and Date blanks. For "Instructor" put your Recitation Instructor's last name. For "Course" put CS 180. For "Test" put Final.

Fill in the bubbles that correspond to your name, section and Student ID in the bubble sheet. For your section number, use 0830, 0930, 1030, 1130, ... – based on the start time of your Friday recitation. Consult the following list:

```
08:30 recitation in LWSN B134: Elizabeth Blythe
09:30 recitation in LILY G401: Matt Carlson
10:30 recitation in LILY G424: Matt Carlson
10:30 recitation in CIVL 1266: Alvin Law
11:30 recitation in REC 122:   Alvin Law
12:30 recitation in LILY G424: Isuru Ranaweera
01:30 recitation in REC 308:   Isuru Ranaweera
02:30 recitation in LWSN B134: Nick Sumner
03:30 recitation in REC 226:   Nick Sumner
```

For your student ID, use the 10 digit ID number on your student ID card. DO NOT USE YOUR SOCIAL SECURITY NUMBER!

Exams without names will be graded as zero. Only the answers on the bubble sheet will be counted. The questions will be discarded.

Recitation Start Time ______________________________

Recitation TA's Name ______________________________

Student Last Name ______________________________

Student First Name ______________________________

Part I. Multiple Choice Questions (3 points each):

1. Which of the following characteristics of an object-oriented programming language restricts behavior so that an object can only perform actions that are defined for its class?

   (a) Dynamic Binding
   (b) Polymorphism
   (c) Inheritance
   (d) Encapsulation ******

2. What is the value of the String `S` after the following line?

```
    String S = (new String("arach")).substring(0,2) +
                    (new String("nophobia")).substring(3);
```

   (a) `"arachobia"`
   (b) `"arnophobia"`
   (c) `"arhobia" ******`
   (d) `"rachobia"`

3. When would you use a private constructor?

   (a) When you get bored with public
   (b) If you want to disallow instantiation of that class from outside that class ******
   (c) If you want to protect your class's members from outside modification
   (d) Never, it's not allowed

4. Which of the following is **true** about RuntimeException and its subclasses?

   (a) If a method throws a RuntimeException, the use of the try/catch block is optional. ******
   (b) The FileIOException class is a subclass of RuntimeException.
   (c) In general, handling of RuntimeException should be done at compile time.
   (d) In general, RuntimeException must be caught with a try/catch block.

5. Which of the following types cannot be used as the parameter for a switch statement?

   (a) char
   (b) boolean ******
   (c) byte
   (d) int

6. What is the output when you try to compile and run the following code?

```
public class Switch {
    public static void main(String[] args) {
        int  i = 1;
        switch( i ) {
            case 0:
                int j = 0;
                System.out.print( j );
            case 1:
                int j = 1;
                System.out.print( j );
            case 2:
                int j = 2;
                System.out.print( j );
            default:
                int j = -1;
                System.out.print( j );
        }
    }
}
```

(a) 12-1
(b) 1
(c) 12
(d) The code does not compile ******

7. 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 );
```

(a) 0
(b) the sum of 100 integers
(c) NullPointerException ******
(d) the code will not compile

8. What is the output of the following code segment?

```
char x = ’A’;
while(x != ’D’){
    switch(x){
        case ’A’:
            System.out.println(x);
            x = ’D’;
        case ’B’:
            System.out.println(x);
            x = ’C’;
            break;
        case ’C’:
            System.out.println(x);
            x = ’D’;
        default:
            continue;
    }
}
```

(a)
```
A ******
D
C
```

(b)
```
A
D
C
D
```

(c)
```
A
D
```

(d) `A`

9. What are valid arguments to the instanceof operator?

(a) a class object and a class type ******
(b) any primitive type
(c) boolean type only
(d) class types only

10. A class which implements the ActionListener interface must implement which method?

(a) void handle(ActionEvent e)
(b) void actionPerformed(ActionEvent e) ******
(c) void eventDispatched(AWTEvent e)
(d) String getActionCommand(ActionEvent e)

11. 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!");
}
```

(a) The catch blocks for exceptions of type B and C are unreachable. ******

(b) A finally block cannot be used with multiple catch blocks.

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

(d) No one loves exceptions and therefore the finally block fails to compile.

12. What is the output of the following program?

```
public class A {
    public static int doStuff(double x, double y) {
        return (int)(x/y);
    }

    public static void main() {
        float x = 6.0;
        int y = 11;

        x = A.doStuff(y,x);

        System.out.print("x="+x+", y="+y);
    }
}
```

(a) `x=1, y=11`

(b) this program does not compile ******

(c) `x=6.0, y=11`

(d) `x=1.0, y=11`

13. 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 );
    ...
```

(a) this code will not compile

(b) Green’s favorite color on Tue Dec 12 10:27:00 EST 2006 is Topato

(c) Topato’s favorite color on Tue Dec 12 10:27:00 EST 2006 is Green

(d) NullPointerException ******

14. After execution of the following code, what will be the values of `x`, `y` and `z`?

```
int x, y, z;
y = 1;
z = 5;
x = 0 - (++y) + z++;
```

(a) `x = 4, y = 2, z = 6`

(b) `x = 4, y = 1, z = 5`

(c) `x = 3, y = 2, z = 6 ******`

(d) `x = -7, y = 1, z = 5`

15. An array object, `ArrayOne`, is created as:

```
float [][] ArrayOne;
ArrayOne = new float[20][10];
```

Suppose `ArrayOne` is passed as an argument to a method in which the corresponding parameter is named `someArray`. What should the declaration of `someArray` look like in the parameter list of the method?

(a) `float [][] someArray ******`

(b) `float someArray[]`

(c) `float [] someArray[20]`

(d) `float someArray[20][10]`

16. 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);
}
```

(a) The Reindeer class does not implement the Serializable interface ******

(b) Only 8 Reindeer objects are written

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

(d) We should use a DataOutputStream instead of an ObjectOutputStream

17. When writing data to a file using a `FileOutputStream`, at what point is the data actually written to the file?

```
  I. Immediately after the write function is called
 II. When the data buffer is full
III. When the close function is called
```

(a) I only

(b) III only

(c) II and III ******

(d) II only

18. Why would a class be declared as abstract?

(a) Because it doesn’t make logical sense to instantiate it ******

(b) So that it can be used as an interface

(c) So that it cannot be inherited from

(d) Because it has no abstract methods

19. Which of the following is **true** about an abstract method inherited into a class C?

(a) It must be defined in C before C can be instantiated ******

(b) None of these is true

(c) It always forces C to become abstract

(d) It overrides any method in C with the same name

20. Based on the class definition below, what can be inferred about the following class `B`:

```
public class B<T extends A> {...}
```

(a) Class T extends B.

(b) B is a bounded parameterized type restricted to be of type T which is of type A or a subclass of A.

(c) T is a bounded parameterized type restricted to be of type A or a subclass of A. ******

(d) Class B extends A.

21. Suppose the class `Undergraduate` extends the class `Student` which extends the class `Person`. Given the following variable declaration:

```
Person p = new Person();
Student s = new Student();
Undergraduate ug = new Undergraduate();
```

Which of the following assignments are legal?

```
  I.  p = ug;
 II.  p = new Undergraduate();
III.  ug = new Student();
 IV.  ug = p;
  V.  s = new Person();
```

(a) III and IV

(b) I and IV

(c) I and II ******

(d) II, III and V

22. Given the following definition of `Bird` and `Chicken`, which of the given statements will not compile?

```
abstract class Bird implements Livestock {}
class Chicken extends Bird {}
```

(a) `Bird bird = new Chicken();`

(b) `Livestock livestock = new Chicken();`

(c) `Bird bird = new Bird(); ******`

(d) None of these will compile

23. Which of the following are **true** regarding the use of generics and parameterized types in Java?

```
  I.  Generics provide type safety by shifting more type checking
      responsibilities to the compiler.
 II.  Generics and parameterized types eliminate the need for downcasts
      when using Java Collections.
III.  When designing your own collections class (say, a linked list),
      generics and parameterized types allow you to achieve type safety
      with just a single class definition as opposed to defining multiple
      classes.
```

(a) I and II
(b) II and III
(c) I, II, and III ******
(d) I and III

24. What type of relationship exists between `someMeth` in classes A and `someMeth` in class B?

```
class A
{
    private void someMeth()
    {
        System.out.println( "from class A" );
    }
}

class B extends A
{
    public void someMeth( String x )
    {
        System.out.println( "from class B: " + x );
    }
}
```

(a) method overriding
(b) method overloading
(c) both method overriding and method overloading
(d) neither method overriding nor method overloading ******

25. Which of the following statements is **true** regarding `Vectors` with no specified parameterized type?

(a) If a parameterized type is not specified the code will not compile
(b) No parameterized type is needed because Java will use the Object class as a parameterized type ******
(c) A parameterized type is needed because Java needs to know how to allocate memory
(d) No parameterized type is needed because Vectors default to storing String objects

26. Given the following definitions, which assignments are legal?

```
    class Box<T>{}
    class SuperBox<T> extends Box<T>{}

    I. Box<Object> b = new Box<String>();
   II. Box<String> b = new SuperBox<String>();
  III Box<Object> b = new SuperBox<String>();
```

(a) I, II, III
(b) II only ******
(c) I and III only
(d) I only

27. Given the function below, what is the value of `f( 8, 9 )`?

```
    private int f( int x, int y ) {
        if( x == 0 ) {
            return y;
        } else {
            return f( x - 1, y + 1 );
        }
    }
```

(a) 0
(b) 17 ******
(c) This recursion is incorrect in some way.
(d) 72

28. Given the function below, what is the value of `g( 3 )`?

```
    private int g( int num ) {
        if( num <= 1 ) {
            return 1;
        } else {
            return 3*g( num-1 ) + g( num-2 );
        }
    }
```

(a) This recursion is incorrect in some way.
(b) 43
(c) 4
(d) 13 ******

29. Given the functions below, what is the sequence: `f(1), f(2), f(3), f(4),` ...

```
int f( int num ) {
    if ( num <= 2 )
        return num;
    else
        return num*g(num-1);
}

int g( int num ) {
    if ( num <= 2 )
        return num;
    else
        return (num-1)*f(num+1);
}
```

(a) `1,2,6,24,...`

(b) This recursion is incorrect in some way. ******

(c) `1,2,3,6,...`

(d) `1,2,6,16,...`

30. what is the output of a call to the printNums() method? Assume inFile has been properly linked to the file whose content is shown below and outFile has been properly linked to some output file.

```
15
23
21
19
```

```
public static void printNums()
{
    int n;
    String line;
    line = inFile.readLine();
    if (line != null) {         //If not EOF ..
        n = Integer.valueOf(line).intValue();
        outFile.print(n + " ");
        printNums();
        outFile.print(n + " ");
    }
}
```

(a) `15 23 21 19 19 21 23 15 ******`

(b) `19 21 23 15 19 21 23 15`

(c) `15 23 21 19`

(d) `19 21 23 15`

The version of your test is **A**. Please **FILL IN CIRCLE (A) for the TEST FORM field on the BUBBLE SHEET** directly under the DATE field and turn in your exam booklet and answer sheet to the stack labeled (A) in the front of the classroom. Thank you.

This page is left blank intentionally.

Part II. Programming Questions (110 points total):

1. (20 points) Write a class, call it **GradesCount**, to read a list of grades from the keyboard (integer numbers in the range 0 to 100). Prompt the user with “Please enter a grade between 0 to 100 or -1 to quit: ” each time before reading the next integer. Store each grade in a A, B, C, D or F Vector as follows: 90 to 100 = A, 80 to 89 = B, 70 to 79 = C, 60 to 69 = D, and 0 to 59 = F. (Hint: You cannot store ints as Vector elements, but you can store Integers.)

   Output the total number of grades entered, the number of A, B, C, D and F, and a list of the A’s. For example, if the input is...

```
38
86
92
55
83
42
90
-1
```

   then the output should be:

```
Total number of grades = 7
Number of A = 2
Number of B = 2
Number of C = 0
Number of D = 0
Number of F = 3
The A grades are: 92, 90
```

Solution for programming question 1:

```
import java.util.*;

public class GradesCount {
    private static int LOW_A = 90, LOW_B = 80, LOW_C = 70, LOW_D = 60;

    public static void main(String[] args) {
        Vector<Integer> A = new Vector<Integer>();
        Vector<Integer> B = new Vector<Integer>();
        Vector<Integer> C = new Vector<Integer>();
        Vector<Integer> D = new Vector<Integer>();
        Vector<Integer> F = new Vector<Integer>();
        int count = 0;
        Scanner scanner = new Scanner(System.in);

        System.out.print("Please enter a grade between 0 to 100 or -1 to quit: ");
        int grade = scanner.nextInt();
        while (grade != -1) {
            if (grade >= LOW_A)
                 A.addElement(new Integer(grade ));
            else if (grade >= LOW_B)
                 B.addElement(new Integer(grade ));
            else if (grade >= LOW_C)
                 C.addElement(new Integer(grade ));
            else if (grade >= LOW_D)
                 D.addElement(new Integer(grade ));
            else
                 F.addElement(new Integer(grade ));
            count++;
            System.out.print("Please enter a grade between 0 to 100 or -1 to quit: ")
            grade = scanner.nextInt();
        }

        System.out.println("Total number of grades = " + count);

        System.out.println("Number of A = " + A.size());
        System.out.println("Number of B = " + B.size());
        System.out.println("Number of C = " + C.size());
        System.out.println("Number of D = " + D.size());
        System.out.println("Number of F = " + F.size());

        System.out.print("The A grades are:");
        if (A.size() > 0)
        {
            for (count = 0; count < A.size()-1; count++)
                System.out.print((Integer)A.elementAt(count) + ", ");
            System.out.print((Integer)A.elementAt(count));
        }
        System.out.println((Integer)A.elementAt(count));
    } // end main
} // end GradesCount
```

2. (20 points) Santa's list gets longer every year and he is having trouble keeping track of what he is going to bring each boy and girl. This year he has gone high-tech and kept a list of names along with whether the child was good or bad on his computer. However, he now needs to know how many toys and how many lumps of coal he needs to buy and for which children. He has hired you to write a program that will read in his list (it's saved in a file called List.txt) and produce another text file called ShoppingList.txt that he can use for delivering gifts and to give the elves so they know how to pack his toy bag.

   The List.txt file is formatted so that each line has the following format:

   ```
   Gender FirstName LastName Status
   ```

   Gender will be either an "F" or a "M" and Status will be either "Good" or "Bad"

   An example List.txt file would be formatted as follows:

   ```
   M Jack Frost Bad
   F CindyLou Who Good
   M Rudolph Rednose-Reindeer Good
   ```

   The ShoppingList.txt file should be formatted such that each line has "lastName, firstName toyName" where toyName is "Coal" if the status of the child is "Bad", toyName is "Pony" if the gender of the child is "F" and the status of the child is "Good", and the toyName should be "Bicycle" if the gender of the child is "M" and the status of the child is "Good". In addition, the last three lines of the file should print the number of lumps of coal, bicycles, and ponies to buy.

   An example resulting ShoppingList.txt file for the above List.txt should be:

   ```
   Frost, Jack Coal
   Who, CindyLou Pony
   Rednose-Reindeer, Rudolph Bicycle

   Lumps of Coal: 1
   Bicycles: 1
   Ponies: 1
   ```

   In order to do this you should create a complete class called **SantasHelper** which will read List.txt and produce ShoppingList.txt. Do not make any assumptions about the length of the list, but you can assume that it is formatted correctly. If the List.txt file does not exist, you should catch any exception that might be thrown.

Solution for programming question 2:

```
import java.util.*;
import java.io.*;

class SantasHelper{
   int numPonies;
   int numBicycles;
   int numLumps;

   SantasHelper(){
      numPonies = 0;
      numBicycles = 0;
      numLumps = 0;
   }

   public static void main(String args[]) throws IOException{
      SantasHelper sh = new SantasHelper();

      try{
         sh.readList();
      }
      catch(FileNotFoundException e){
         System.out.println(e.getMessage());
      }
   }

   public void readList() throws IOException, FileNotFoundException{

      String lastName, firstName, status, gender;
      File list;
      File outFile;
      int i = 0;

      list = new File("List.txt");
      outFile = new File("ShoppingList.txt");

      FileOutputStream outFileStream = new FileOutputStream(outFile);
      PrintWriter outStream = new PrintWriter(outFileStream);

      Scanner scanner = new Scanner(list);

      while(scanner.hasNext()){
         gender = scanner.next();
         firstName = scanner.next();
         lastName = scanner.next();
         status = scanner.next();
         if(status.equals("Bad")){
             outStream.println(lastName + ", " + firstName + " Coal");
             numLumps++;
         }
         else if(gender.equals("F")){
             outStream.println(lastName + ", " + firstName + " Pony");
             numPonies++;
         }
         else{
             outStream.println(lastName + ", " + firstName + " Bicycle");
             numBicycles++;
         }
      }

      outStream.println("\nLumps of Coal: " + numLumps);
      outStream.println("Ponies: " + numPonies);
      outStream.println("Bicycles: " + numBicycles);
      outStream.close();
      scanner.close();
   }
}
```

3. (20 points) Several design properties for a problem are presented below. Use these properties in order to write all the necessary classes and/or interfaces for a solution to the problem. Focus on class structure and interaction. You may implement your solution however you wish, but you will be graded on the appropriateness of your solution to the requirements. Note the use of capitalization and parentheses for clarification. You may use whatever constructors or additional methods you wish.

   You must define a structure that can represent Animals. Animals have two behaviors; they can speak() and they can move(). By default, when an animal moves, the text ”This animal moves forward” is displayed. By default, when an animal speaks, the text ”This animal speaks” is displayed. A general Animal should not be able to be instantiated.

   Define also two classes, Goose and Lynx, that are Animals. Both Goose and Lynx behave such that where ”animal” is displayed in speak() or move(), ”goose” or ”lynx” is displayed by the appropriate classes.

   Finally, any instance of Goose can fly(), just as any Flying object can. An Airplane is also a Flying object. Define the Airplane class such that it is Flying and make sure that any instance of Goose is also Flying. The specific behaviors when instances of either class fly() are up to you. Instances of either Goose or Airplane should be able to be stored in a variable of type Flying.

Solution for programming question 3:

```
abstract class Animal
{
    private final String name;
    public Animal( String name )
    {
        this.name = name;
    }
    public Animal()
    {
        this( "animal" );
    }
    public void speak()
    {
        System.out.println( "This " + name + " speaks" );
    }
    public void move()
    {
        System.out.println( "This " + name + " moves forward" );
    }
}

class Lynx extends Animal
{
    public Lynx()
    {
        super( "lynx" );
    }
}

interface Flying
{
    public void fly();
}

class Goose extends Animal implements Flying
{
    public Goose()
    {
        super( "goose" );
    }
    public void fly()
    {
        System.out.println( "This " + getClass().getName() +
                            " soars, wings flapping.");
    }
}
```

```
class Airplane implements Flying
{
    public void fly()
    {
        System.out.println( "This " + getClass().getName() +
                            " soars, engines running.");
    }
}
```

4. (25 points) PART 1: In computer science, a Stack is a LIFO (last in, first out) data structure. Objects most recently inserted into a stack are the first objects removed from the stack. One way to design a stack data structure is to build it as a chain of node objects (like a linked list). The first part of this question involves designing an appropriate node class. Your node class should take advantage of parameterized types such that any type of object can be placed inside the node. Minimally, you should have appropriate class variables, constructors, and a set of accessors/mutators for each of your class variables. Fill in the Node class on the next page:

Solution for programming question 4 part 1:

```
  public class Node<T> {
    // add your class variables here
    private T item;
    private Node<T> next;

    // add your constructors and methods here
    public Node<T>(T item, Node<T> next) {
        setItem(item);
        setNext(next);
    }

    public Node<T> getNext() { return next; }
    public void setNext(Node<T> next) { this.next = next; }

    public T getItem() { return item; }
    public void setItem(T item) { this.item = item; }
}
```

PART 2: Now, write a class Stack. Your class should have a parameterized type such that any type of item can be placed in the Stack. You should also use your Node class as defined above to hold the contents of the stack. Minimally, your Stack class must have the following:

- A Node object which points to the top of the stack
- A default constructor which sets up an empty Stack (this can be represented with a null Node object)
- An insertion method, push, which should take in a parameterized object and add it to the top of the stack.
- A deletion method, pop, which has no parameters. Pop should remove the top node from the stack and return the element inside it (not the entire node). If the stack is empty, return a null object.
- A peeking method, peek, which simply returns the top element on the stack. If the stack is empty, return a null object.

Solution for programming question 4 part 2:

```
public class Stack<T> {

    private Node<T> stack;

    public Stack<T>() { stack = null; }

    public T pop() {
        if (stack == null) return null;
        T popped = stack.getItem();
        stack = stack.getNext();
        return popped;
    }

    public void push(T item) {
        stack = new Node<T>(item,stack);
    }

    public T peek() {
        if (stack == null) return null;
        return stack.getItem();
    }
}
```

5. (25 points) A palindrome is a word or phrase that reads the same forward or backwards. Write a recursive method that returns a boolean value indicating if its only String argument is a palindrome or not. The method must be recursive and have the following signature:

```
public static boolean isPalindrome( String arg )
```

In testing for palindrome you should ignore upper/lower case as well as whitespace. That is, the following phrases should be legal palindromes.

```
No Sir prefer prison
Racecar
Straw Warts
never odd or even
Oozy rat in a sanitary zoo
```

Place the isPalindrome method in a class named Palindrome. Write a main method in Palindrome which prompts for a string from the keyboard, test if it is a palindrome, and print out the result. Below is a sample output.

```
The string: ’No Sir prefer prison’ is a palindrome
The string: ’Try this’ is NOT a palindrome
```

Solution for programming question 5:

```
import java.util.*;

public class Palindrome
{
    public Palindrome()
    {
    }

    public boolean isPalindrome(String s)
    {
        if (s.length() <= 1)
           return true;    // base case
        else
        {
            if (s.toLowerCase().charAt(0) ==
                s.toLowerCase().charAt(s.length() - 1))
                return isPalindrome(s.substring(1, s.length() - 1));
            else
                return false;
        }
     } // end isPalindrome()

    public static void main(String[] args)
    {
        Palindrome p = new Palindrome();
        String s, target;
        Scanner scan = new Scanner(System.in);
        String lineBreak = System.getProperty("line.separator");
        scan.useDelimiter(lineBreak);

        System.out.println("Please enter a string for testing:");
        s = scan.next();
        target = s.replace(" ", "");
        if (p.isPalindrome(target))
        {
            System.out.println("The string: '" + s +"' is a palindrome");
        }
        else
        {
            System.out.println("The string: '" + s +"' is NOT a palindrome");
        }
    }

} // end Palindrome
```