1. **Short answer (20 points):** Write short (1-2 sentence) answers to the questions below.

1. **What does JVM stand for?**
   Java Virtual Machine

2. **What are the two main components of a node in a linked list?**
   Data, and a “link” to the next node in the list.

3. **Give one advantage of using a linked list to store data compared to using an array.**
   You can easily add as many nodes as you like to a linked list, and you can easily add a node to any location in the list.

4. **Give one advantage of using an array to store data compared to using a linked list.**
   You can easily access any specific entry of the array, whereas this is more difficult with a linked list.

5. **What is the return type of the BufferedReader method read()?**
   int

6. **What is the return type of the Vector method elementAt(int i)?**
   Object

7. **Why might a programmer declare a method to be final?**
   A final method can’t be overridden in a derived class, so a programmer would declare a method to be final if he or she didn’t want that method to ever be overridden.

8. **From inside a derived class, can you directly access private instance variables of the base class? Explain the reason why Java works this way and not the other way.**
   No. If you could, anyone willing to go to the extra trouble of creating a derived class could access your private variables, representing a significant loss of security (control) to the programmer.

9. **What is the signature of the method you must write when you implement the ActionListener interface?**
   ```java
   public void actionPerformed(ActionEvent e)
   ```
10. Which sorting method that we studied would be the best (most efficient) one to use on an array that is almost sorted.

   Bubble sort.

2. True/False (10 points): Circle T if the statement is true and F if the statement is false.

   T  F  The Java compiler creates code in the machine language of the computer on which it is run.
   T  F  To use the JFrame class, you must import javax.swing.*
   T  F  A final class can not be extended.
   T  F  If the class B extends the class A then an array of A objects can hold objects of type B also.
   T  F  Writing two methods with the same name and return type but with different numbers of arguments is an example of method overriding.
   T  F  The body of a do while loop is executed at least one time.
   T  F  If the class B extends the class A then only one of the two classes can have a method with a given signature.
   T  F  When attempting to initialize a PrintWriter object Java may throw a FileNotFoundException.
   T  F  To implement the cloneable interface, you must provide a method with signature public Object clone().
   T  F  The default layout manager for the content pane of a JFrame object is the FlowLayout.

3. Exceptions (10 points): Look at the blocks of code below and write the type of exception (or “None”) that would be thrown if the code were executed.

   ```java
   int[] a = new int[10];
   a[9] = 5;
   int i = 9;
   System.out.println(a[i++]);
   ```

   Exception:  _____None_____________________
int[] a = new int[10];
for(int i = 0; i <= 10; i++)
{
    a[i] = i;
}

Exception:  ____ArrayIndexOutOfBoundsException_______

String[] s = new String[5];
s[3] = new String("64 slices of American cheese");
int j = 3;
System.out.println(s[--j]);

Exception:  _____None_______________

Warrior zantar = new Warrior();
Warrior temp = (Warrior) zantar.clone();

Exception:  _____CloneNotSupportedException_____

String[][] t = new String[5][5];
int i = 0, j = 0;
while(i<5)
{
    j = 0;
    while(j<5)
    {
        t[i][j] = ""+(char)(i*j);
        j++;
    }
    i++;
}

Exception:  _______None___________
4. **Strings (10 points):** Write the body of the method provided below. A palindrome is a string that reads the same forwards or backwards. For example, “abba” and “Jay is si yaJ” are both palindromes but “Hi mom” and “abbA” are not (so your method is case-sensitive). Your method should return true if the input string is a palindrome and false otherwise. Don’t forget good programming style!

```java
public boolean isPalindrome(String sentence)
{
    //The largest int less than half the number of chars in //the sentence:
    int half = sentence.length()/2;

    int length = sentence.length();

    boolean isAPalindrome = true;

    //Check to see if the string is the same forwards & backwards
    for(int i = 0; i <= half; i++)
    {
        if(sentence.charAt(i) != sentence.charAt(length - 1 - i))
        {
            isAPalindrome = false;
        }
    }

    return isAPalindrome;
}
```

```java
public boolean isPalindrome(String sentence)
{
    //The largest int less than half the number of chars in //the sentence:
    int half = sentence.length()/2;

    int length = sentence.length();

    boolean isAPalindrome = true;

    //Check to see if the string is the same forwards & backwards
    for(int i = 0; i <= half; i++)
    {
        if(sentence.charAt(i) != sentence.charAt(length - 1 - i))
        {
            isAPalindrome = false;
        }
    }

    return isAPalindrome;
}
```
5. **Sorting (15 points):** Consider the arrays below and answer the following questions.

```java
int[] a = {5, 4, 9, 0, 2, -3, 25, 1};
```

On the three lines below, write the state of the array `a` after each of the first three passes of the bubble sort algorithm (a “pass” is one time through the outer loop of the algorithm).

- After pass 1:   4    5    0    2    -3    9    1    25
- After pass 2:   4    0    2    -3   5    1    9    25
- After pass 3:   0    2    -3    4    1    5    9    25

```java
String[] s = {“rev”, “jay”, “ben”, “aaron”, “darius”, “chihiro”, “alvin”};
```

On the three lines below, write the state of the array `a` after each of the first three passes of the selection sort algorithm (a “pass” is one time through the outer loop of the algorithm).

- After pass 1:   aaron, jay, ben, rev, darius, chihiro, alvin
- After pass 2:   aaron, alvin, ben, rev, darius, chihiro, jay
- After pass 3:   aaron, alvin, ben, rev, darius, chihiro, jay

How many total comparisons would be made if the array `a` above (starting from the original unsorted state) was completely sorted with the selection sort algorithm?

28
6. **Recursion (10 points):** Write the body of the method below. Your method should be recursive (no credit will be given for a non-recursive solution), and should calculate the sum of the numbers 1, 2, ..., n. For example, if your method is called with the input \( n = 5 \) then the result should be 15. You may assume that the input to your method will always be positive. Your code should easily fit inside the space provided and you should not write any code outside of the method definition. Don’t forget good programming style!

```java
public int sumOfInts(int n)
{
    // base case: sum of ints between 1 and 1 is 1
    if(n == 1)
    {
        return 1;
    }
    else // recursive step
    {
        return n + sumOfInts(n-1);
    }
}
```

//Use the space below to draw some diagrams if you like.
7. OOP and Inheritance (15 points): In the space on the next two pages (use the backs of the pages if necessary), write two new classes. The first class is the class Triangle and should have three private instance variables of type double. Conceptually, these three variables will hold the lengths of the three sides of your triangle. Provide a default constructor and a constructor which allows for the input of the three side lengths. Write a method isEquilateral to take no input and return true if the triangle is an equilateral triangle and false otherwise. You may use == when comparing doubles. Write a final method perimeter to return a double representing the perimeter of the triangle. Write a method isValidTriangle to return true if the three side lengths would allow for the formation of a triangle and false otherwise. (You can’t, for example, have a triangle with side lengths 1, 2, and 10. The side lengths a, b, and c must satisfy the triangle inequality: a + b >= c for each labeling of the sides (a,b,c) in order to be able to form a triangle.)

Now write the class EquilateralTriangle to be a child class of the parent class Triangle. Your default constructor should create an equilateral triangle with unit length sides, and you should provide a constructor that takes a single side length as input and creates an appropriate equilateral triangle. Override the method isEquilateral to always return true and do the same for the method isValidTriangle. Write a new method area to take no input and return a double representing the area of your equilateral triangle. The height of an equilateral triangle with side length ‘a’ is a*sin(60) = a*(0.866) (approximately).

```java
public class Triangle {
    private double a, b, c; // lengths of the sides

    // default constructor creates an equilateral triangle with side length one
    public Triangle() {
        a = 1; b = 1; c = 1;
    }

    // non-default constructor
    public Triangle(double side1, double side2, double side3) {
        a = side1; b = side2; c = side3;
    }

    // an accessor method for side length a:
    public double getA() {
        return a;
    }

    public boolean isEquilateral() {
        // if all three sides are equal, it’s equilateral
    }
}```
if(a==b && b==c)
{
    return true;
}
return false;
}
public final double perimeter()
{
    return a + b + c;
}

public boolean isValidTriangle()
{
    boolean isValid = true;
    if(a + b <= c)
        isValid = false;
    if(a + c <= b)
        isValid = false;
    if(b + c <= a)
        isValid = false;
    return isValid;
}
}

public class EquilateralTriangle extends Triangle
{
    //Default constructor
    public EquilateralTriangle()
    {
        super(1,1,1);  //in my solution super() would work also
    }

    //non-default constructor
    public EquilateralTriangle(double sidelengths)
    {
        super(sidelengths, sidelengths, sidelengths);
    }

    public boolean isEquilateral()
    {
        return true;  //of course it’s an equilateral triangle!
    }

    public boolean isValidTriangle()
    {
        return true;  //ditto
    }

    public double area()
    {
        return (0.5)*0.866*getA();  //one-half base times height
    }
}
8. Procedural programming and arrays (10 points): The stub of a method is provided for you below. Write code inside the method definition so that the method returns an array of booleans of the specified length. The boolean entry at index i should be true if the integer \((i+2)\) is prime, and false if the integer \((i+2)\) is composite. A positive integer \(n > 1\) is prime if its only positive divisors are 1 and \(n\). Your code should fit on this page.

```java
public boolean[] isPrime(int length) {
    boolean[] primes = new boolean[length];

    //Loop through the numbers 2 through (length + 2) to check if prime:
    for(int i = 0; i < length; i++)
    {
        int num = i + 2;  //number to check
        primes[i] = true;

        //see if num is divisible by any of the numbers 2 through (num – 1)
        for(int j = 2; j < num; j++)
        {
            if(num%j==0)  //if num is divisible by j, it’s not prime.
            {
                primes[i] = false;
                break;  //exit the loop, it’s already not prime =(
            }
        } // end inner loop
    } //end outer loop

    return primes;
}
```
9. **GUI programs (20 points):** Look at the window below. Write Java code that would display this window on the screen. Use the front and back of this page. You must make the “Exit” button functional (the program should end if the user presses the exit button.) The window has height 200, width 400, and uses a GridLayout.

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SimpleGUI extends JFrame implements ActionListener
{
    private JLabel myLabel = new JLabel("What is your name?");
    private JTextField myField = new JTextField(20);
    private JButton exitButton = new JButton("Exit");

    public SimpleGUI()
    {
        super();
        setSize(400,200);
        setTitle("A simple GUI");

        Container contentPane = getContentPane();
        contentPane.setLayout(new GridLayout(2,1));

        JPanel p1 = new JPanel();
        JPanel p2 = new JPanel();
        p1.add(myLabel);
        p1.add(myField);
        p2.add(exitButton);
        contentPane.add(p1);
        contentPane.add(p2);

        exitButton.addActionListener(this);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```
public void actionPerformed(ActionEvent e)
{
    if(e.getActionCommand().equals("Exit"))
    {
        System.exit(0);
    }
}

public static void main(String[] args)
{
    SimpleGUI a = new SimpleGUI();
    a.setVisible(true);
}
}