Answer all questions in the space provided. Use the backs of the pages if you need extra space. You may have a single-page one-side crib sheet for this exam.

If a question asks for code and you’re not sure how to write the code, write comments to explain the part you don’t know how to write. It is better to write something showing what you know than to leave questions blank.

**There are no deliberate errors in the code on this exam.**

Please make sure that you have all 10 pages before beginning the test.

If you are stuck on a question, move on and come back to it later.

<table>
<thead>
<tr>
<th>No.</th>
<th>Pts.</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
1. (18 points) Give an example of each of the following items.

a. A declaration of a constant class variable (one value for all members of the class)

   ```java
   public static final double CENTIMETERSPERINCH = 2.54;
   ```

   (can be public or private, but must be static and final, and have an initial value).

b. A call to a non-static (instance) method. You do not have to show the declaration, only the call.

   ```java
   point1.getX();  // It should be clear that the call is based on an object, not a class, that is, not “Point.getX()”.  
   ```

c. A local variable declaration. Include enough context to make it clear that the variable is local.

   ```java
   public void f(){
      int x;  // this variable is local in f
      ... }
   ```

d. A statement block.

   ```java
   {
      x = 1;  // one or more statements within braces is a block
   }
   ```

e. A method that is overloaded. Include enough context to make it clear that the method is overloaded.

   ```java
   void method (int x, int y){  // Same name & different parameter lists
      // first definition
   }

   void method (double d){
      // second definition
   }
   ```

f. An appropriate use of the keyword this.

   ```java
   Point () {
      this(0, 0);  // calls 2-parameter constructor with default values
   }

   Point(x, y){
      this.x = x;  this.y = y;
   }
   ```
2. (10 points) What is printed when main is called in the following class?

```java
import java.awt.*;
public class problem2 {
    public static void f(int x, int y) {
        System.out.println("f: x is "+ x + " y is "+ y);
        int tmp = y;
        y = -x;
        x = tmp;
        System.out.println("f: x is "+ x + " y is "+ y);
    }

    public static void g(Point p) {
        System.out.println("g: x is "+ p.x + " y is "+ p.y);
        int tmp = p.y;
        p.y = -p.x;
        p.x = tmp;
        System.out.println("g: x is "+ p.x + " y is "+ p.y);
        f(p.x, p.y);
        System.out.println("g: x is "+ p.x + " y is "+ p.y);
    }

    public static void main(String[] args) {
        int a = 15;
        int b = 20;
        Point p1 = new Point(1,2);
        System.out.println("main: a is "+ a + " b is "+ b);
        f(a,b);
        System.out.println("main: a is "+ a + " b is "+ b);
        f(b,a);
        System.out.println("main: a is "+ a + " b is "+ b);
        g(p1);
        System.out.println("main: p1.x is "+ p1.x + " p1.y is "+ p1.y);
    }
}
```

```
main: a is 15 b is 20
f: x is 15 y is 20
f: x is 20 y is -15
main: a is 15 b is 20
f: x is 20 y is 15
f: x is 15 y is -20
main: a is 15 b is 20
```

```
g: x is 1 y is 2
f: x is 2 y is -1
f: x is 2 y is -1
main: p1.x is 2 p1.y is -1
```
3. (16 points) In this question, you will write code that works with arrays.
   
   a. Write a declaration of an array of 4 strings. The name of the array should be GroceryList, and the contents of the list should be: “Eggs”, “Milk”, “Cereal”, and “Coke”.

   ```java
   String[] GroceryList = new String[] { “Eggs”, “Milk”, “Cereal”, “Coke” };
   ```

   b. Write a statement that will replace “Cereal” by “Trix” within the array.

   ```java
   ```

   c. Write a code fragment that will make GroceryList an empty array of 5 elements.

   ```java
   GroceryList = new String[5];
   ```

   d. Write a code fragment that creates a scanner, reads 5 strings, and places them (in the order entered) in GroceryList. For full credit, your code fragment must contain a for loop.

   ```java
   Scanner stdin = new Scanner(System.in);
   for(int i=0;(i<5)&&stdin.hasNext();i++){
       GroceryList[i] = stdin.next();
   }
   ```
4. (16 points) This question asks you to use the Collection class, ArrayList to solve almost the same problem as question 3.

a. What is one advantage of an ArrayList over an array?

An ArrayList will grow or shrink to accommodate items being added or removed from the list.

b. Write a declaration of an ArrayList of 4 strings. The name of the array should be GroceryList, and the contents of the list should be: “Eggs”, “Milk”, “Cereal”, and “Coke”.

```java
ArrayList<String> GroceryList = new ArrayList<String>();
GroceryList.add("Eggs");
GroceryList.add("Milk");
GroceryList.add("Cereal");
GroceryList.add("Coke");
```

c. Write a statement that will replace “Cereal” by “Trix” within the array.

```java
GroceryList.set(2, "Trix");
```

d. Write a code fragment that creates a scanner, reads 5 strings, and places them (in the order entered) in GroceryList. For full credit, your code fragment must contain a for loop.

```java
Scanner stdin = new Scanner(System.in);
for(int i=0;(i<5)\&\&stdin.hasNext();i++){
    GroceryList.add(stdin.next());
    //or GroceryList.set(i, stdin.next());
}
```
5. (20 points) Create a class called RandomTest. Your class will have only a main function and any necessary instance variables.

a. Write the class, including a main function that generates 10 random numbers between 1 and 10 (it should be possible to generate 1 or 10, but not 0 or 11) and adds them up and prints the result. The output should be something like “Your random total is 85.” Make sure that your program is complete, including any necessary import statements.

```java
import java.util.Random
public class RandomTest
{
    public static void main (String[] args)
    {
        Random generator = new Random();
        int sum = 0;
        for(int i=0;i<10;i++)
        {
            //random number gets 1 added because it would
            //be 0-9 otherwise
            sum = sum + generator.nextInt(10)+1;
        }
        System.out.println("Your random total is "+sum;
    }
}
```

b. Modify the main function so that instead of generating exactly 10 numbers, it asks the user for an integer, generates that number of random numbers and adds them up and prints the result. The output should be something like “Your random total of 10 values is 85.” You do not need to rewrite the parts that don’t change as long as you are very clear about what parts do change.

```java
import java.util.Random
import java.util.Scanner //extra line (or just import java.util.*)
public class RandomTest
{
    public static void main (String[] args)
    {
        Random generator = new Random();
        int sum = 0;
        Scanner stdin = new Scanner(System.in);
        System.out.print("How many numbers ?");
        int count = stdin.nextInt();
        //Changed 10 to count below
        for(int i=0;i<count;i++)
        {
```
{ 
    //random number gets 1 added because it would
    //be 0-9 otherwise
    sum = sum + generator.nextInt(10)+1;
}

//Modified output statement to include count
System.out.println("Your random total of " + count +" values is " + sum;
6. (10 points) In our lab about dates, we computed the differences between dates, but did not implement date comparison.

a. Write a equals() method for the Date class. Here is some example code to show you how equals should work:

```java
Date deadline = new Date (2007, 2, 28);
Date today = new Date (2007, 3, 1);
if (today.equals(deadline))
    System.out.println(“Today is the deadline!”);

boolean equals(Object another){
    if (another instanceof Date){
        return ((this.year == another.year) &&
                (this.month == another.month) &&
                (this.day == another.day))
    }
    else return false;  //another is not even a Date
}
```

b. Create a predicate method isBefore in the Date class. (A predicate method is one that returns a boolean value). You may use the Date class as it was defined in your lab. Here is some example code to show you how isBefore should work:

```java
Date deadline = new Date (2007, 2, 28);
Date today = new Date (2007, 3, 1);
if (deadline.isBefore(today))
    System.out.println(“You are past the deadline!”);

public Boolean isBefore(Date other){
    //If the years are different, use them to decide
    if(this.year != other.year)
        return (this.year < other.year);
    //The year is the same
    //If the months are different, use them to decide.
    if(this.month != other.month)
        return (this.month < other.month);
    //The month is also the same, use the day
    return (this.day < other.day);
}
6. (10 points) In the following class, the variable n occurs in multiple scopes. Each time n is used (except for a declaration), draw an arrow to the applicable declaration of n.

Comments in /* */ indicate definitions. Comments after // represent “arrows” pointing to the like-numbered definitions

```java
public class x()
{
    public x()
    {
        n = “N”; //→ D5
    }
    public int f()
    {
        int n = 1; /*D1*/
        return n; //→ D1
    }
    public int g (int k)
    {
        int a;
        for (int n  = 1; /*D2*/ n <= k; n++) //→ D2 (for both)
            a = a + n; //→ D2
        System.out.println(n); //→ D5
        return a;
    }
    public int k (int n /*D3*/)
    {
        if (n < 0) //→ D3
        {
            int m = -n; //→ D3
        }
        return n*m; //→ D3
    }

    string n /*D5*/;
}
```
(Extra page for scratch paper)