Name: _____

CS 141: Introduction to (Java) Programming: Exam 3

Jenny Orr • Willamette University • Fall 2013

1.	(max 8)	5.	(max 18)
2.	(max 6)	6.	(max 12)
3.	(max 9)	7.	(max 18)
4.	(max 6)	8.	(max 23)
Total:			(max 100)

1. (2 pts each, 8 pts total) What is the value of either ans or result after each of the statements below. Assume the declarations:

```
double ans = 2.0;
int result = 1;
```

a. `result = 11 % 4;` _____3_____

b. `result = 4 % 10;` _____4_____

c. `result = 10 / 4;` _____2_____

d. `ans  = 3/10 + 7.5;` _____7.5_____

2. (2 pts each, 6 pts total) What is the value (true or false) of each of the following Boolean expressions assuming

```
int x = 10;
```

Circle correct answer

a) `x != -1 || x == 10` **True** or False or Incorrect Syntax

b) `!( 20 < x  < 70)` True or False or **Incorrect Syntax**

c) `!(x == 4 && x > 5)` **True** or False or Incorrect Syntax

3. (3 pts each, 9 total) What is the value of the following code snippets?

   a. Snippet:

```
int sum = 0;
int n = 1;
while (n < 7)
{
  sum = sum + n;
  n = n + 2;
}
System.out.println("sum = " + sum);
```

Output: `sum = 9`

sum	n
0	1
1	3
4	5
9	7

---

   b. Snippet:

```
int cnt = 0;
for (int j = 0; j < 50; j++) {
   for (int i = 0; i < 10; i++) {
      cnt++;
   }
}
System.out.println("cnt = " + cnt);
```

Output: `cnt = 500`

50 x10 = **500**

---

   c. Snippet:

```
String greet = "Happy New Year!";
String greet2 = greet.substring(1,8);
System.out.println(greet2);
```

Output: `appy Ne`

4. (6 pts total) 1D Arrays:

   a. (2 pts) Write code that declares and creates a 1-dimensional array of doubles called dNums. The length of the array is 10.

```
double [] dNums = new double[10];
```

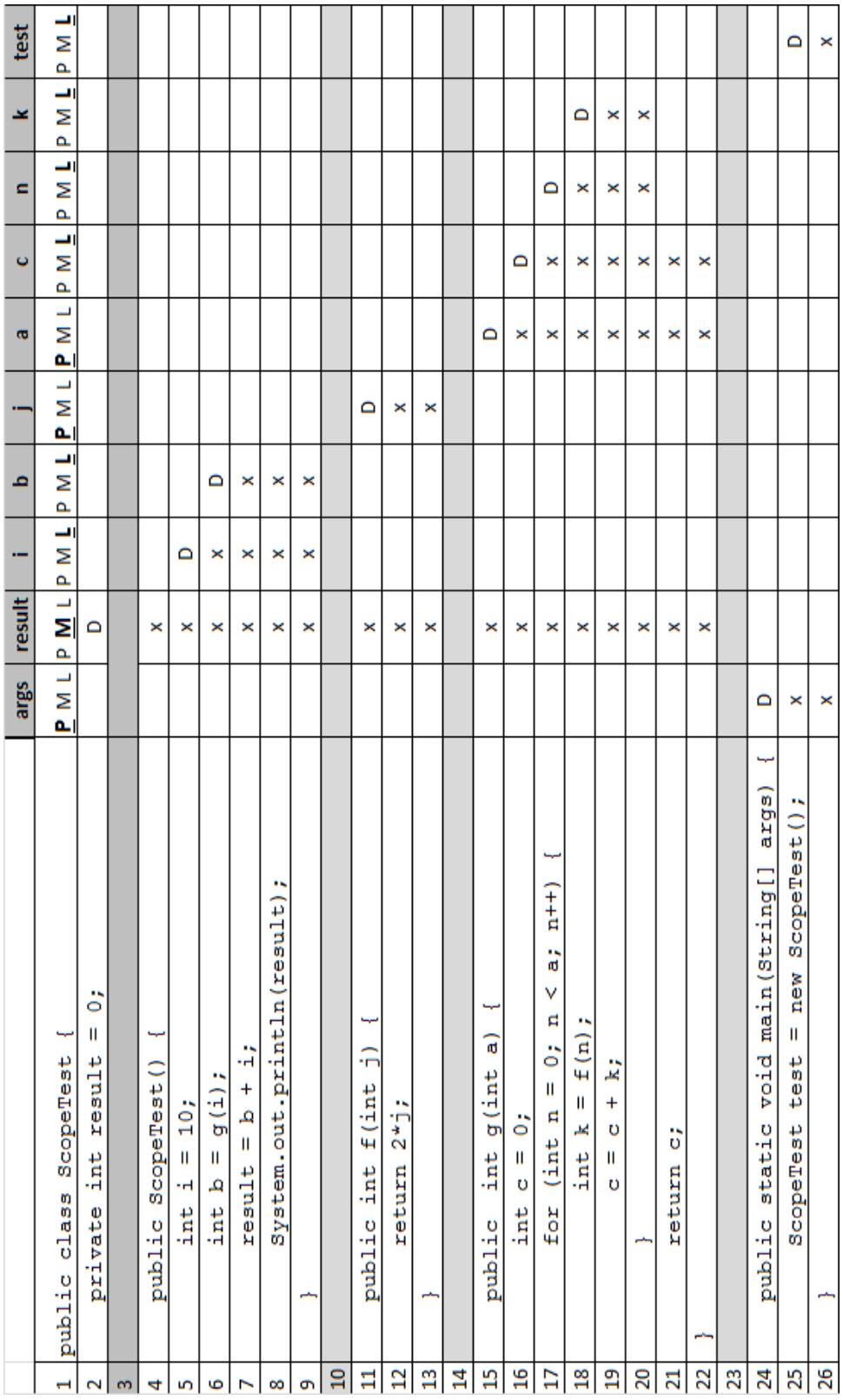
   b. (4 pts) Write a for-loop which loads the array with random numbers in the range 0 to 1.

```
for (int i = 0; i < dNums.length; i++) {

        dNums[i] = Math.random();

}
```

5. (18 pts) For each variable name listed at top, indicate a "D" on the line where the variable is declared. Mark an x in boxes to indicate scope. Circle "P" at the top if the variable is formal parameter, "M" if it is a member variable and "L" if it is a local variable.



		args				result				i				b				j				a				c				n				k				test		
		P	M	L		P	M	L		P	M	L		P	M	L		P	M	L		P	M	L		P	M	L		P	M	L		P	M	L	
1	public class ScopeTest {																																							
2	private int result = 0;					D																																	
3																																							
4	public ScopeTest() {								x																															
5	int i = 10;								x			D																												
6	int b = g(i);								x			x				D																								
7	result = b + i;								x			x				x																								
8	System.out.println(result);								x			x				x																								
9	}								x			x				x																								
10																																								
11	public int f(int j) {								x									D																						
12	return 2*j;								x									x																						
13	}								x									x																						
14																																								
15	public int g(int a) {								x													D																		
16	int c = 0;								x													x				D														
17	for (int n = 0; n < a; n++) {								x													x				x				D										
18	int k = f(n);								x													x				x				x				D						
19	c = c + k;								x													x				x				x				x						
20	}								x													x				x				x				x						
21	return c;								x													x				x														
22	}								x													x				x														
23																																								
24	public static void main(String[] args) {				D																																		
25	ScopeTest test = new ScopeTest();				x																																D			
26	}				x																																		x	

6. (12 pts total) 2-Dimensional Arrays: Suppose `myNums` is a 2-dimensional array of integers with 5 rows and 8 columns.
   a. (2 pts each, 4 pts total) What is the ***value*** of each of the following

`myNums.length` _____5_____

`myNums[2].length` _____8_____

   b. (2 pts each, 8 pts total) What is the ***type*** of each of the following where the type should be selected from the list below :
      A. 1-D array of doubles
      B. 1-D array of ints
      C. 2-D array of Strings
      D. 2-D array of ints
      E. ArrayList of Strings
      F. ArrayList if ints
      G. int
      H. String
      I. double

Circle the answer:

`myNums[0]`	A	**B**	C	D	E	F	G	H	I
`myNums[2][3]`	A	B	C	D	E	F	**G**	H	I
`myNums.length`	A	B	C	D	E	F	**G**	H	I
`myNums`	A	B	C	**D**	E	F	G	H	I

7. (18 pts) Classes:
   a. (15 pts) Create a Vehicle class consisting of the following:
      - Two private instance variables for the vehicle's type (String) and year (int).
      - A constructor which sets the value of both instance variables.
      - A setter & getter for year.
      - A toString method
      - The code should be consistent with the following declaration and print statement:

```
Vehicle vehicle = new Vehicle ("truck", 2010);
System.out.println(vehicle);
```

Which should output: `Type: truck, Year: 2010`

```
public class Vehicle {

    // Private instance member variables for type and year:

    private String type;
    private int year;

    // Constructor:

    public Vehicle(String type, int year) {
        this.type = type;
        this.year = year;
    }

    // Getter and Setter for year


    public int getYear() {
        return year;
    }

    public void setYear(int y) {
        year = y;
    }

    // toString

    public String toString() {
        return "Type: " + type + ", Year: " + year;
    }

}
```

b. (3 pts) Suppose you want to create a subclass of `Vehicle` called `Car` which has an additional private instance `String` variable called `make` (e.g. whose value could be `"Ford"`). Below, write only the ***constructor*** for `Car`. It should be consistent with the declaration:

```
Car car = new Car("car", 2010, "Ford");
```

// Constructor:

```
public Car(String type, int year, String make) {
    super(type, year);
    this.make = make;
}
```

8. (1 pt each, 23 pts total) **True and False**: Please circle T or F

2.1) **T or F:** Object parameters are passed by reference.

2.2) **T or F:** The keyword word `static` is used to indicate instance methods and variables.

2.3) **T or F**: A class can implement more than one interface.

2.4) **T or F**: A class cannot have more than one superclass.

2.5) **T or F** In an abstract class, not all of the methods are implemented.

2.6) **T or F:** If one changes the value of a class variable, the value is *not* changed for all objects of that type.

2.7) **T or F:** If Car is a subclass of Vehicle, then it is ok to have the declaration:

```
Car c = new Vehicle();
```

2.8) **T or F:** An object may be created from an abstract or concrete class.

2.9) **T or F:** The keyword `super` is used to call the constructor of a superclass.

2.10) **T or F**: To create a subclass, one uses the `implements` keyword.

2.11) **T or F**: A subclass has access to protected instance variables of its superclass.

2.12) **T or F**: In the RGB color format, c represents black if

```
Color c = new Color(255,255,255);
```

2.13) **T or F**: When a user presses a button, an event is generated.

2.14) **T or F**: A superclass inherits data and behavior from a subclass.

2.15) **T or F**: GUI components such as JButtons can be found in the Swing library.

2.16) **T or F:** It is never ok for two methods in a class to have the same name.

2.17) **T or F:** An anonymous inner class can access to the member variables of the enclosing class.

2.18) **T or F:** The `catch` part of a try-catch is used to indicate what to do if no errors are generated.

2.19) **T or F:** A constructor should always have a `void` return value.

2.20) **T or F:** An class's constructor is called when the keyword `new` is used.

2.21) **T or F:** Once an object is garbage collected, it can still be retrieved if needed again.

2.22) **T or F:** Private *member variables* can be accessed outside of the class using setters and getters.

2.23) **T or F:** Suppose `setValue` is a method with one parameter of type int. When *calling* the method, you need to provide an *actual parameter*, e.g. `setValue(3)`. And when *declaring* the method, you need to provide a *formal parameter*, e.g., `setValue(int n)`.