Name: ____________________________

CS 141: Introduction to (Java) Programming: Exam 1

Jenny Orr • Willamette University • Spring 2018

SOLUTIONS

1.	(max 10)	4.	(max 12)	7.	(max 14)
2.	(max 10)	5.	(max 18)	8.	(max 12)
3.	(max 3)	6.	(max 9)	9.	(max 12)
Total:		(max 100)			

1. (2 pts each, 10 pts total) Assume the variables have the following types and that their values are set somewhere else in the program:

```
char letter;                          String street1;
double x;                             String street2;
double y;
```

Write Boolean expressions that test for each of the conditions below:

a) `letter` is equal to the ascii character a. ______ `letter =='a'`__________

b) x has a value between 0 and 100. ______ `0 < x && x < 100`_______________

c) `street1` and `street2` reference the same String object. ___ `street1 == street2`__

d) `street1` and `street2` contain the same sequence of chars. `street1.equals(street2)`

e) neither x nor y are smaller than 100. `x >= 100 && y >= 100` or `!(x<100 || y<100)`

2. (2 pts each, 10 pts total) Given variables below, indicate whether the following Boolean expressions evaluate to true or false by circling the correct answer or indicating that the expression has an error :

```
int n = 6;                            double b = 7.9;
double a = 2.5;                       boolean bool = true;
```

a) `(a > 0 && bool)` [✓ true] false error

b) `!( a < 2.1 || b > 10)` [✓ true] false error

c) `( n % 10 == 2)` true [✓ false] error

d) `( a <= b <=  2*a )` true false [✓ error]

e) `( n/20 == 0)` [✓ true] false error

3. (3 pts) Suppose x and y are both integers. The following expression

```
! ( !(x <= 10) || (y == 7))
```

is equivalent to which of the following choices below (circle your choice). Make use of what you know about simplifying Boolean expressions, including DeMorgan's Laws.

a) `x <= 10  || y != 7`

[✓ b) `x <= 10  &&  y != 7`]

c) `x >= 10  || y == 7`

d) `x >= 10  &&  y == 7`

4. (4 pts each, 12 pts total) Each code snippet below contains a coding error. Explain the error and show one way to rewrite the code to fix the problem.
the error.

a.
```
double d = 23;
int n = d;
```
ans: Can't store double in int

Fix:
```
double d = 23;
double n = d;
```
or
```
double d = 23;
int n = (int) d
```

b.
```
Die d;
int r = d.roll();
```
ans: No object created

Fix:
```
Die d = new Die();
int r = d.roll();
```

c.
```
// assume s is a String declared and set elsewhere in code.
if (s = "Sam") {
     s = s + " Jones";
}
```
ans: Condition uses assignment statement and not equality

Fix:
```
if (s == "Sam") {
    s = s + " Jones";
}
```

or
```
if (s.equals("Sam")) {
    s = s + " Jones";
}
```

5. (1 pts each, 18 pts total) **True and False:** Please circle T or F
(Credit is only given if the instructor can clearly tell which answer is circled)

1) **[✓ T] or F:** Java will garbage collect an object if the object has no references pointing to it.

2) **[✓ T] or F:** A boolean variable has only 2 possible values.

3) **[✓ T] or F:** The name of a constructor must be the name of the class.

4) **[✓ T] or F:** Primitive variables are passed to methods by value rather than by reference.

5) **T or [✓ F]:** Division has a higher precedence than multiplication.

6) **[✓ T] or F:** A `char` is a primitive type while a `String` is a class type.

7) **[✓ T] or F:** An object's *member* variables exist for as long as the object exists.

8) **T or [✓ F]:** A return statement can return multiple variables.

9) **[✓ T] or F:** By convention, method and variable names should begin with a lower case letter while class names should begin with an upper case letter.

10) **[✓ T] or F:** The name of a class must always be the same as the java file where it is defined.

11) **T or [✓ F]:** In the code: `String s = new String("hello");`
we call "hello" the parameter.

12) **[✓ T] or F:** An error detected by the compiler that is a violation of the program language rules is called a syntax error.

13) **T or [✓ F]:** An object stores its data in methods.

14) **[✓ T] or F:** Each *object* of a class has its own set of member variables.

15) **[✓ T] or F:** A method header consists of: an access specifier, a return type, a method name, and a list of the parameters (if any).

16) **[✓ T] or F:** A constructor is executed when an object is created.

17) **[✓ T] or F:** A member variable is a variable which is declared in the class but outside of any method.

18) **[✓ T] or F:** When a method is done executing, local variables in the method variables no longer exist.

6. (9 pts) Write a multi-way if-else statement that prints `"Your grade is A"` when the test score is 90 or above, `"Your grade is B"` if the test score is less than 90 but greater than or equal to 80, and `"Your grade is C"` otherwise. Assume the test score is stored in a double called `score`.

   You only need to write the if-else statement. You do not need to write any other surrounding code.

```
if (score >= 90) {

    System.out.println("Your grade is A");

} else if (score >= 80) {

    System.out.println("Your grade is B");

} else {

    System.out.println("Your grade is C");

}
```

7. (14 pts total) Scope: For each variable listed at the top, indicate the following:

   a. On the row below the variable name, indicate whether the variable is a
      - parameter (**P**)
      - local variable (**L**), or
      - member variable (**M**).

   b. Place a **D** on the line where the variable is declared.

   c. Mark an **x** to indicate the variable's scope.

Notes: The variable `args` is already completed as an example.
Not all variables in the code are listed at top.

		args	perc	bk	price	t	profit	p	reduce
	Enter P, L or M →	**P**	**L**	**L**	**M**	**P**	**L**	**P**	**L**
1	`public class BookProject{`								
2	`  public static void main(String[] args) {`	D							
3	`     double perc = 4.2;`	x	D						
4	`     Book bk = new Book("Intro to Java",perc);`	x	x	D					
5	`  }`	x	x	x					
6	`}`								
7	`public class Book {`				x				
8	`  private String title;`				x				
9	`  private double price = 54.0;`				D				
10	`  private int sold = 240193;`				x				
11	`  public Book(String t, double perc) {`				x	D			
12	`    title = t;`				x	x			
13	`    double profit = calcProfit(perc);`				x	x	D		
14	`    System.out.print(title + " has profit:");`				x	x	x		
15	`    System.out.println("  $" + profit);`				x	x	x		
16	`  }`				x	x	x		
17	`  public double calcProfit(double p) {`				x			D	
18	`    System.out.println("Calculating profit");`				x			x	
19	`    double pro = price*sold*p/100.;`				x			x	
20	`    if (sold < 1000) {`				x			x	
21	`       double reduce = 0.75 ;`				x			x	D
22	`       pro = pro*reduce;`				x			x	x
23	`    }`				x			x	x
24	`    return pro;`				x			x	
25	`  }`				x			x	
26	`}`				x				

8. (12 pts total) Classes:
   a) Create a `Restaurant` class that includes the following:
      - Member variables: `name` (e.g. `"Ram Brewery"`) and `price` (which is the average price of one meal e.g. 12.50) (*use the* *<u>appropriate</u>* *data types*).
   - One constructor which takes two inputs for setting the values of *each* member variable.
   - A setter & getter for the `name` variable.
   - A method called `mealPrice` which takes as input a single variable for the number of people at a meal (e.g. called `numPeople`), and returns the average price of the meal for that many people.
   - A `toString` method which, for the above values, might output something like:

```
Restaurant: Ram       Average price: $12.50
```

(don't worry about insuring that only 2 decimal places are printed).

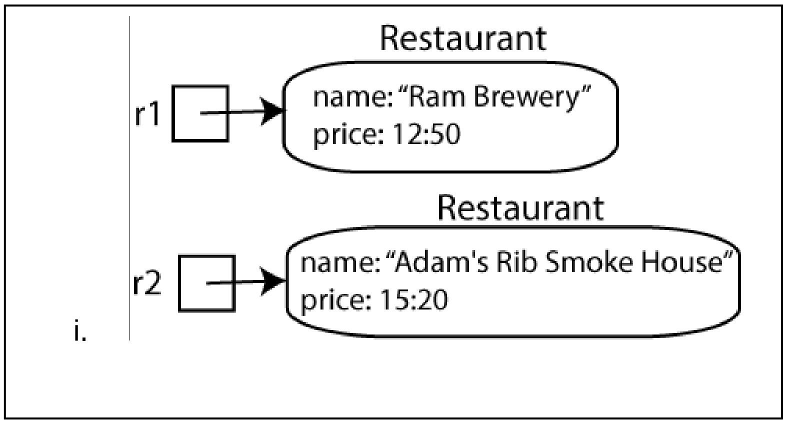
```
public class Restaurant {
    // Member variables:

     private String name;
     private double price;

    // Constructor:

     public Restaurant(String n, double p) {
         name = n;
         price = p;
     }

   // Getter for name:

    public String getName() {
        return name;
    }

   // Setter for name:

    public void setName(String n) {
        name = n;
    }

   // mealPrice method

     public double mealPrice(int numPeople) {
         return price*numPeople;
     }

   // toString:

    public String toString() {
        return "Restaurant: " + name +  "    Average Price: $" + price;
    }
}
```

9. (12 pts total) Unit Testing:
   a) (6 pts) Based on the definition of the Restaurant class in the previous problem, in main:
      1) Create two `Restaurant` objects, e.g. called `r1` and `r2`.
      2) Use the methods, e.g. `mealPrice` and the *getter*, to compute and print the cost of a meal for 4 people at one of your restaurants. The output might look like:

```
A meal for 4 at the Ram Brewery will cost around $50.00
```

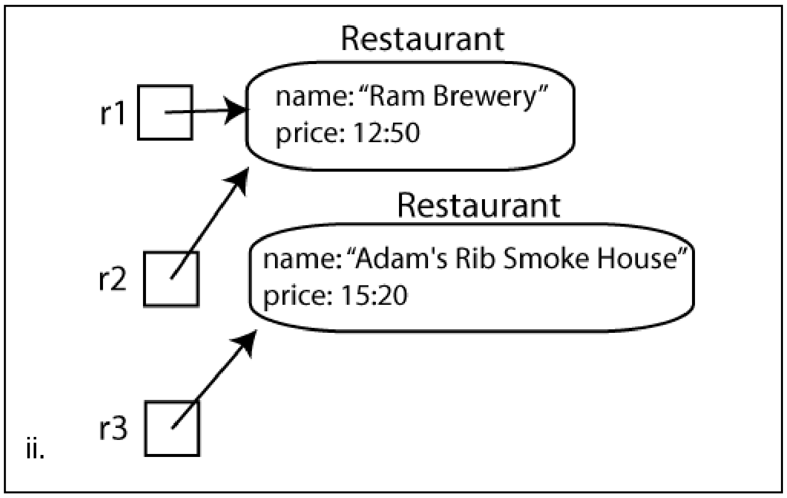
```
public class RestaurantUnitTest {
    public static void main(String[] args) {

     // part 1: create 2 Restaurant objects:

        Restaurant r1 = new Restaurant("Ram Brewery", 12.50);
        Restaurant r2 = new Restaurant("Adam's Rib Smoke House", 15.20);

    // part 2: Use the methods in the class to calculate and print the
    // cost of a meal for 4 at one of your restaurants.

        System.out.println("A meal for 4 at the " + r1.getname() +
                  " will cost around $" + r1.mealPrice(4));

    }
}
```

   b) (6 pts) Memory object diagrams:
   i. Based on the code above, draw in the box on the right the memory (references and objects) immediately after the two `Restaurant` objects have been created (i.e. after part 1 has executed).



   ii. Draw how the memory would look if we added the instructions after part 1:

```
Restaurant r3 = r2;
r2 = r1;
```


Restaurant
r1
name: "Ram Brewery"
price: 12:50
Restaurant
r2
name: "Adam's Rib Smoke House"
price: 15:20
r3
ii.