

**Com S 227**  
**Spring 2019**  
**Exam 1x**

**DO NOT OPEN THIS EXAM UNTIL INSTRUCTED TO DO SO**

Name: Shounak Lahiri

ISU username: Sblahiri@iastate.edu

Lab section/TA name: (please circle one)

*Your exam will be returned to you in the section you select.*

	M	T	W	R	F
8:00 – 10:00			(14) Jose	(8) Alvin, Diego	
10:00 – 12:00		(12) June, Kamini	(4) Logan, Jeff K.	(5) Alvin, Oyendrila	
12:00 – 2:00			(7) Jacob, Shaiqur	(1) Oyendrila, June	(13) Ethan R., Zoe
2:00 – 4:00		(15) Ethan V.	(9) Sovann, Marcia	(6) Waqwoya, Long	
4:00 – 6:00		(3) Mailin, Jay	(2) Jack, Jeff Y.		
6:00 – 8:00			(11) Yonas, Alex	(10) Waqwoya,, Evan	

Closed book/notes, no collaboration, no electronic devices, no headphones. Time limit 70 minutes. Partial credit may be given for partially correct solutions. *If you have questions, please ask!*

- There are some excerpts from the Java API documentation on the last page
- Use correct Java syntax for writing code.
- You do not need comments or import statements.
- It is ok to use literal numbers in your code
- You can write "SOP" to abbreviate "System.out.println"

Question	Points	Your Score
1	20	19
2	15	15
3	10	10
4	30	30
5	10	10
6	5	3
7	10	10
Total	100	97



1. (20 pts) a) Assume the declarations at right. For each expression in the left-hand column below, give its *value* and its *type* (int, double, boolean, String, etc.) in the second and third columns. If what's written is not a valid Java expression, just write "error" in both columns. (The first one is done for you as an example.)

```
int k = 17;
double d = 2.5;
String name = "george";
Rectangle r1 = new Rectangle(1,2,3,4);
Rectangle r2 = new Rectangle(1,2,3,4);
```

Expression	Value	Type
d * k	42.5	double
k / 3	5	integer
name.length() != k	true	Boolean
name.length() = k	Error	Error
k % 3	2	integer
k.length() == 2	Error	Error
!(k > 0 && d > 5) (k < 0 && d < 5)	true	Boolean
r1 == r2;	False	Boolean

- (b) - (d) Complete the right-hand side of each assignment by writing an expression satisfying the stated requirement. (In some cases there is more than one correct answer. Do not write any additional lines of code, just write an expression.) The first one is done for you.

(Example) A boolean value indicating whether the int variable **x** is positive

```
boolean isPositive = x > 0;
```

- (b) A boolean value indicating whether the int variable **x** is an even number

```
boolean isEven = x % 2 == 0;
```

- (c) Given an int value **x**, a boolean indicating whether **x** is greater than 12 and less than 17

```
boolean isInSpecifiedRange = (x > 12 && x < 17);
```

- (d) Given a int value **h** of hours and **m** of minutes, a double that is the total amount of time in hours (e.g. 30 minutes is 0.5 hours)

```
double totalTime = (h * 1.00) + (m / 60.0);
```

19



2. (15 pts) The Speedy Deluxe Pizza company sells large pizzas for 8.00 each, or you can get 5 for 30.00. Normally there is a delivery fee of .50 per pizza, but delivery is free to ISU students. Write an interactive main method that prompts the user to enter the number of pizzas and a "yes" or "no" indicating whether or not they are an ISU student, and prints the total cost of the order including delivery. A sample interaction is shown below, with user's responses in **bold**. Note that the price of 49.50 is correct for the inputs shown. It is ok to assume that anything other than the exact string "yes" can be interpreted as a "no".

```
import java.util.Scanner;
public class SomeClass
{
```

```
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        System.out.println("How many? ");
        String pizzas = in.next+Int();
        System.out.println("ISU student (yes/no)? ");
        String student = (in.next()).toLowerCase();

        double price = 0.00;
        if ( (student.equals("yes")) == false)
        {
            price += pizzas * .50;
        }
        if ( (pizzas / 5) > 0)
        {
            price += (pizzas / 5) * 30.00;
            pizzas = pizzas % 5;
        }
        if ( pizzas > 0)
        {
            price += pizza * 8.00;
        }
        System.out.println("Price: " + price);
    } // End method
} // End class
```

```
How many? 7
ISU student (yes/no)? nah
Price: 49.5
```

Problems 3 and 4 refer to a class named **GumballMachine**. See overview below. Detailed specification in Javadoc form is at the top of the next page.

In problems 3 and 4 we will refer to a class named **GumballMachine** that models an old-fashioned gumball machine. Its API documentation is shown below. A gumball machine contains a certain number of gumballs, up to a fixed maximum. It has a handle with a slot. A coin can be inserted into the slot. If the handle is turned when a coin is in the slot, the coin goes inside the machine and a gumball comes out, if possible. (That is, if the machine is empty, the coin still goes in, but the number of gumballs never goes below zero.) Turning the handle when there is no coin in the slot has no effect. More gumballs can be added to refill the machine. Assume that the machine can hold an unlimited number of coins.





(Before beginning problems 3 and 4, see the overview in the box on the previous page)

### Constructor and Description

**GumballMachine** (int givenMax)

Constructs a gumball machine that can hold the given maximum number of gumballs. Initially the machine is empty and contains no coins.

Modifier and Type	Method and Description
void	<b>turnHandle()</b> Simulates turning the handle on the gumball machine. If there is a coin in the slot, the coin goes into the machine, and the number of gumballs is reduced by 1 if possible. If there is no coin in the slot, this method does nothing.
void	<b>add(int howMany)</b> Adds the given number of gumballs to the machine, if possible (the number of gumballs in the machine cannot go over the maximum value specified in the constructor).
int	<b>getNumGumballs()</b> Returns the number of gumballs currently in the machine. This value is never negative.
int	<b>getTotalCoins()</b> Returns the total number of coins currently in the machine.
void	<b>insertCoin()</b> Inserts a coin into the slot. If there is already a coin in the slot, this method does nothing.

3. (10 pts) Assume that you have a class **GumballMachine** with the constructor and methods given in the Javadoc above. Write a short main method that will test whether the **GumballMachine** implementation is working correctly for the following three cases (for each case, your output should print the *actual* result and the *correct* result, no other output is required):

1. For a machine constructed with capacity 100 gumballs, after add(75) twice, it should contain 100 gumballs.
2. For a machine containing 100 gumballs, after insertCoin() and turnHandle(), it should contain 99 gumballs and one coin.
3. Turning the handle when no coin is present should not change the number of gumballs.

**public class GumballMachineTest**      Assume you do not need any import statements.

```

{
    public static void main(String[] args)
    {
        GumballMachine g = new GumballMachine(100);
        g.add(75);
        g.add(75);
        System.out.println("Expected: 100 ; Actual: " + g.getNumGumballs());
        g.insertCoin();
        g.turnHandle();
        System.out.println("Expected: 99 ; Actual: " + g.getNumGumballs());
        System.out.println("Expected: 1 ; Actual: " + g.getTotalCoins());
        g.turnHandle();
        System.out.println("Expected: 99 ; Actual: " + g.getNumGumballs());
        System.out.println("Expected: 1 ; Actual: " + g.getTotalCoins());
    }
} // End main

```

3 // End class



(Before beginning problems 3 and 4, see the overview in the box on page 3!)

4. (30 pts.) Write the complete class `GumballMachine` according to the Javadoc on the previous page. Remember that all instance variables should be **private** and the constructor and methods should be **public**.

```
30
public class GumballMachine
{
    private int maxGum;    // Max gumballs
    private int gum;       // Current number of gumballs
    private int totCoins;  // Total number of coins
    private boolean coin;  // whether there is a coin in the slot

    public GumballMachine(int givenMax)
    {
        maxGum = givenMax;
        gum = 0;
        totCoins = 0;
        coin = false;
    }

    public void turnHandle()
    {
        if (coin == true)
        {
            totCoins += 1;
            coin = false;
            gum = Math.max((gum - 1), 0);
        }
    }

    public void add(int howMany)
    {
        gum = Math.min((gum + howMany), maxGum);
    }

    public int getNumGumballs()
    {
        return gum;
    }

    public int getTotalCoins()
    {
        return totCoins;
    }

    public void insertCoin()
    {
        coin = true;
    }
} //End class
```

5. (10 pts) Write a static method `guessThatPrice` that uses an instance of `Random` to guess the price of a prize in a quiz show. The price is a minimum of 3 dollars, and is a maximum of 24 dollars and 99 cents. The method should return a string of the form "*d* dollars and *c* cents" where *d* is a random value 3 through 24, and *c* is a random value from 0 through 99.

```
import java.util.Random;
public class SomeClass
{
    public static String guessThatPrice()
    {
        // TODO
        Random rand = new Random();
        int d = (rand.nextInt(21)) + 3;
        int c = rand.nextInt(100);
        return d + "dollars and " + c + "cents";
    } // End method
} // End class
```



6. (5 pts) Rewrite the `foo` method at right so that it does exactly the same thing but does not use any conditional statements (no "if" or "if/else" statements, ternary expressions, while statements, etc.). (Partial credit may be given if you do it with just one conditional statement.)

(3) `if (x > 0 && x > y)`  
`{`  
`return true;`  
`}`  
`else`  
`{`  
`return false;`  
`}`

```
public boolean foo(int x, int y){
    if (x > 0){
        if (y > x){
            return false;
        }
        else{
            return true;
        }
    }
    else {
        return false;
    }
}
```

$\max(x, y)$   $\rightarrow$   $(x > y \text{ ? } x : y)$   
 $\rightarrow$  `return true;`

7. (10 pts) A class `Elvis` is shown at right. The class `Exam1` below has a main method:

```
public class Exam1
{
    public static void main(String[] args)
    {
        Elvis r1 = new Elvis(3);
        Elvis r2 = new Elvis(5);
        Elvis r3 = r2;
        x=3 System.out.println(r1.rock(7));  $\rightarrow 6$ 
        x=3 System.out.println(r1.rock(-1));  $\rightarrow -2$ 
        x=100 System.out.println(r1.rock(7));  $\rightarrow 200$ 
        r3.roll(42);
        System.out.println(r2.rock(7));  $\rightarrow 84$ 
    }
}
```

In the box below, write down the output printed on the console when the `Exam1` main method is run.

6  
-2  
200  
84

```
public class Elvis
{
    private int x;

    public Elvis (int givenX)
    {
        x = givenX;
    }

    public void roll(int givenX)
    {
        x = givenX;
    }

    public int rock(int y)
    {
        int z = 2 * y;
        if (z < 0)
        {
            x = 100;
        }
        else
        {
            z = 2 * x;
        }
        return z;
    }
}
```

	X	return		X	return
1 <sup>st</sup>	3	6	2 <sup>nd</sup>	5	N/A
2 <sup>nd</sup>	3	-2	3 <sup>rd</sup>	42	84
3 <sup>rd</sup>	100	200			