# Lab 5: Chapter 5, “Writing Classes” SOLUTION

The following exercises are intended to help you apply and practise the concepts introduced in this module. This work is **not** submitted for marks.

Try to answer the questions on paper first. Then insert the code in a Java program to see the actual result.

1. For each of the following pairs, which represents a class and which represents an object of that class?
2. Superhero, Superman
3. Justin, Person
4. Rover, Pet
5. Magazine, Time
6. Christmas, Holiday

**SOLUTION**

1. Superhero (C), Superman (O)
2. Justin (O), Person (C)
3. Rover (O), Pet (C)
4. Magazine (C), Time (O)
5. Christmas (O), Holiday (C)
6. List some attributes and operations that might be defined for a class called Meeting that represents a business meeting.

**SOLUTION**

attributes start date/time, end date/time, location, attendees

operations get\_ and set\_ for start, end, location; add attendee, del attendee

1. List some attributes and operations that might be defined for a class called Course that represents a college course (not a particular offering of a course, just the course in general).

**SOLUTION**

attributes code, title, credits, pre-requisite(s)

operations get\_ and set\_ for code, title, credits; add pre-requisite, del pre-requisite

1. Write a method called cube that accepts one integer parameter and returns that value raised to the third power.

**SOLUTION**

static int cube (int num)

{

return (num \* num \* num);

}

public static void main(String[] args)

{

System.out.println( cube(2) );

System.out.println( cube(3) );

System.out.println( cube(5) );

}

1. Write a method called random100 that returns a random integer in the range of 1 to 100 (inclusive).

**SOLUTION**

static int random100 ()

{

Random rng = new Random();

return (rng.nextInt(100) + 1);

}

public static void main(String[] args)

{

// enough tests that you should see 1 and 100 in the output

for (int i = 1; i <= 300; i++)

System.out.println(random100());

}

1. Write a method called alarm that prints the string “Alarm!” multiple times on separate lines. The method should accept an integer parameter that specifies how many times the string is printed. Print an error message if the parameter is less than 1.

**SOLUTION**

static void alarm (int n)

{

if (n < 1)

System.out.println ("Error, not a positive number");

for (int i = 1; i <= n; i++)

System.out.println ("Alarm!");

}

public static void main(String[] args)

{

alarm(-3);

alarm(5);

}

1. Write a method called countA that accepts a String parameter and returns the number of times the character ‘A’ is found in the string.

**SOLUTION**

static int countA (String str)

{

int k = 0;

for (int i = 0; i < str.length(); i++)

if (str.charAt(i) == 'A')

k++;

return k;

}

public static void main(String[] args)

{

System.out.println (countA("ALLANA"));

}

1. Write a method called evenlyDivisible that accepts two integer parameters and returns true if the first parameter is evenly divisible by the second, or vice versa, and otherwise false. Return false if either parameter is zero.

**SOLUTION**

static boolean evenlyDivisible (int i, int j)

{

if ( ( i == 0 ) || ( j == 0 ) )

return false;

if ( ( i%j == 0 ) || ( j%i == 0 ) )

return true;

return false;

}

public static void main(String[] args)

{

System.out.println (evenlyDivisible(1,0)); //false

System.out.println (evenlyDivisible(0,1)); //false

System.out.println (evenlyDivisible(15,3)); //true

System.out.println (evenlyDivisible(3,15)); //true

System.out.println (evenlyDivisible(18,5)); //false

System.out.println (evenlyDivisible(5,18)); //false

}

1. Write a method called multiConcat that takes a String and an integer as parameters. Return a String that consists of the string parameter concatenated with itself count times, where count is the integer parameter. For example, if the parameter values are “hi” and 4, the return value is “hihihihi”. Return the original string if the integer parameter is less than 2.

**SOLUTION**

static String multiConcat (String str, int count)

{

if (count < 2)

return str;

String strConcat = str;

for (int i = 2; i <= count; i++)

strConcat = new String (strConcat + str);

return strConcat;

}

public static void main(String[] args)

{

System.out.println (multiConcat("hi",-2));

System.out.println (multiConcat("hi",4));

}

1. Overload the multiConcat method from Exercise 5.25 such that if the integer parameter is not provided, the method returns the string concatenated with itself. For example, if the parameter is “test”, the return value is “testtest”.

**SOLUTION**

static String multiConcat (String str)

{

String strConcat = str + str;

return strConcat;

}

public static void main(String[] args)

{

System.out.println (multiConcat("test"));

}

1. Draw a UML class diagram for the CountFlips program.

**SOLUTION**



This was drawn using Microsoft Visio. Some notations are slightly different from the examples in the textbook.