North South University
Department of Electrical and Computer Engineering
CSE 215L: Programming Language II Lab
Lab Manual - 15
Lab Instructor: Taif Al Musabe

Objective:
- Practice OOP

Task - 1:
A Java interface can only contain method signatures and fields. The interface can be used to achieve polymorphism. In this problem, you will practice your knowledge on interfaces.

You are given an interface *AdvancedArithmetic* which contains a method signature *int divisor_sum(int n*). You need to write a class called *MyCalculator* which implements the interface.

divisorSum function just takes an integer as input and return the sum of all its divisors. For example divisors of 6 are 1, 2, 3 and 6, so *divisor_sum* should return 12. The value of n will be at most 1000.

Read the partially completed code in the editor and complete it. You just need to write the *MyCalculator* class only. Your class shouldn't be public.

Input:	**Output:**
6	I implemented: AdvancedArithmetic 12

Explanation: Divisors of 6 are 1,2,3 and 6. 1+2+3+6=12.

Task - 2:
When a subclass inherits from a superclass, it also inherits its methods; however, it can also override the superclass methods (as well as declare and implement new ones). Consider the following *Sports* class:

```
class Sports{
    String getName(){
        return "Generic Sports";
    }
    void getNumberOfTeamMembers(){
        System.out.println( "Each team has n players in " +
getName() );
    }
}
```

Next, we create a *Soccer* class that inherits from the Sports class. We can override the *getName* method and return a different, subclass-specific string:

```
class Soccer extends Sports{
    @Override
    String getName(){
        return "Soccer Class";
    }
}
```

Complete the code in your editor by writing an overridden *getNumberOfTeamMembers* method that prints the same statement as the superclass' *getNumberOfTeamMembers* method, except that it replaces with (the number of players on a Soccer team).

Formatted Output
Generic Sports Each team has n players in Generic Sports Soccer Class Each team has 11 players in Soccer Class

Homework:

A Java abstract class is a class that can't be instantiated. That means you cannot create new instances of an abstract class. It works as a base for subclasses. You should learn about Java Inheritance before attempting this challenge.

Following is an example of abstract class:

```
abstract class Book{
    String title;
    abstract void setTitle(String s);
    String getTitle(){
        return title;
    }
}
```

If you try to create an instance of this class like the following line you will get an error:

```
Book new_novel=new Book();
```

You have to create another class that extends the abstract class. Then you can create an instance of the new class.

Notice that *setTitle* method is abstract too and has no body. That means you must implement the body of that method in the child class.

In the editor, we have provided the abstract *Book* class and a *Main* class. In the *Main* class, we created an instance of a class called *MyBook*. Your task is to write just the *MyBook* class.Your class mustn't be public.

Sample Input	**Sample Output**
A tale of two cities	The title is: A tale of two cities