Introducing a new object-oriented complexity metric

Sewvandi Wickramasinghe

TABLE OF CONTENTS

Contents

INTRODUCTION

In the modern era, Object Oriented language has become a successful language surmounting other language. Most of the object-oriented languages provide the service of inheritance, encapsulation, data hiding, polymorphism, abstraction, aggregation. Etc. Therefore, it increases the ability of software to reuse, test, maintain and extend.[1]

The following proposed new object-oriented (OO) complexity metrics is invoked based on the knowledge gathered from Cyclomatic Complexity (CC) and Weighted Composite Complexity (WCC) metrics. In order to introduce a new complexity metric, the assignment group has selected the following four factors.

- Encapsulation
- Inheritance
- Polymorphism
- Abstraction

The newly proposed complexity metric can be used to measure the complexity of each program statement, line by line, based on the above-mentioned factors. In addition to that, this metric can be used to easily develop a complexity measuring tool.

Four Complexity Factors

1)Encapsulation

Explanation of how encapsulation considered in the new metric affects to the complexity of a program.

- Object-orientated architecture supports encapsulation which avoids unauthorized access to data. Encapsulation is a method to reduce interdependencies between separately written modules through the definition of strict foreign interfaces. Encapsulated complexity class tests two aspects, one is the aspect of encapsulation and one is the aspect of complexity. Class complexity can be managed by maximization of the object design encapsulation. Class complexity metric encapsulated consisting of encapsulated method and attributes and available method and attributes. Getting more classes does not mean more complexity because the number of attributes and methods in classes depends on the complexity of the class architecture. A more complex form with more attributes and methods is challenging to understand. As a result, the complexity of the code can be reduced by increasing the encapsulation.[1]

Explanation of how the new metric captures the complexity introduced by encapsulation.

- Encapsulation criteria is used to classify encapsulated classes. The encapsulated classes can be identified by the getter and setter methods. Also, by the invocation of public methods in the encapsulated class. If the getters and setters are used in the class, the metric parameter value is set to -1. We have considered a minus value for this parameter since encapsulation decreases the complexity of a program. Therefore, the cumulative code complexity coverage factor should also be reduced. All the lines of codes inside a getter method or a setter method implementation are also set to -1. The metric parameter value would be set to 1 (positive value) if there is no usage of encapsulation class since normal repeating statements increases the complexity of a program. Therefore, a plus value will be added to the cumulative code complexity coverage factor.

2)Inheritance

Explanation of how inheritance considered in the new metric affects to the complexity of a program.

- By using inheritance inside a program, we can make the program more reusable. The child classes can reuse the methods and attributes of the parent class by inheriting. The child classes can add or remove certain features of the parent class code such as overriding methods of the parent class inside the child class. The child class does not affect any changes in the parent class when inheriting the methods and the attributes of the parent class. In addition to that, the child classes can have their own methods concurrently with the inherited methods.

- Inheritance usage decreases the required amount of device maintenance and decreases the testing workload. The reuse by inheritance of software appears to be more durable, more understandable and more efficient. These measurements help to identify highly complex classes and programs. This identification can either channel further quality testing efforts or redesign programs to reduce complexity. This process identifies the IS-A relationship between a parent class and a children's class.[2]

Explanation of how the new metric captures the complexity introduced by inheritance.

- The newly proposed metrics identifies the structure of the inheritance as follows.
    - If the class is a parent class, the Q value will be assigned to 0.
    - If the class is a child class which is extended by a parent class, the Q value will be assigned to 1.
    - If the class is a child class which is implemented by one interface, the Q value will be assigned to 2.
    - If the class is a child class which is implemented by two or more interfaces, then the Q value will be counted by the following equation
        - Q = x + 1 (x = number of implementing interfaces)
    - If the class is a child class which is extended by a parent class and implemented by any number of interfaces, then the Q value will be counted by the following equation.
        - Q = 1 + x + 1

- These values will be allocated for the corresponding class’s every line of code due to complexity increment.

3)Polymorphism

Explanation of how polymorphism considered in the new metric affects to the complexity of a program.

- Polymorphism is a key mechanism in object-oriented programming, which enables objects of several classes to answer calls for broader actions defined for the base class. Each class can execute its class-specific procedure if instructed to execute a particular method or return a class member's value. Polymorphism also allows a class to descent from other classes with regard to inheritance, but the subclass may identify particular features that differ from its ancestor. Code which invokes the same method for such objects, depending on its type, can get different results from each object. The specific signatures that they hold in various declarations are recognized through overridden methods. The implementation of polymorphism would diminish the complexity of an object-oriented system.[3]

Explanation of how the new metric captures the complexity introduced by polymorphism.

- According to the newly proposed metric, the complexity due to polymorphism can be addressed as follows.
    - If polymorphism has not occurred in a line of code of a program, then the value for R is considered as 0.
    - If the program contains any overridden methods, that means the polymorphism is occurred in the program. Therefore, the R value is considered as -1 since polymorphism decreases the complexity of a program. This decrement is taken as -1. All the lines of codes inside an overridden method implementation are also set to -1.

4)Abstraction

Explanation of how abstraction considered in the new metric affects to the complexity of a program.

- Abstraction helps to build codes that are well-designed. Abstraction helps in constructing loosely coupled codes. The reason is that abstraction acts as a type and encapsulates the potential changes behind the abstract class / interface.[4] Data abstraction helps one to convert the complicated data structure into one that is quick and easy to use. The effect of this is that a program with a high level of code complexity can be transformed into a program that looks similar to English.[5]

- Abstract methods don't have a body. If a class has an abstract form, it will be considered abstract, vice versa is not valid, which means that an abstract class does not have to have a mandatory abstract form. If a standard class extends an abstract class, the class must either follow the abstract methods of the abstract parent class, or it must also be considered abstract. All interface methods are by default public abstract. You can't have concrete methods in the interface.[6]

Explanation of how the new metric captures the complexity introduced by abstraction

- The newly proposed metrics identifies the structure of the abstraction as follows.
    - If a class is not an abstract class, the S value is considered as 0.
    - If a class is an abstract class, the S value is considered as -1.
    - Interfaces are pure abstract classes. Therefore, the S value for interfaces are also considered as -1.

Above mentioned four key factors have been used as,

- P - Encapsulation factor
- Q - Inheritance factor
- R - Polymorphism factor
- S - Abstraction factor

LOC - Line of Code

P - Encapsulation factor

- If the LOC contains a get method or a set method: P = -1
- Else: P = 1

Q – Inheritance factor

- If a class is a parent class: Q = 0
- If a class extends another class: Q = 1
- If a class implements an interface: Q = 2
- If a class implements 2 or more interfaces: Q = x + 1 (x = number of implementing interfaces)
- If a class extends a parent class and implements interfaces: Q = 1 + x + 1

R - Polymorphism factor

- If the LOC is an overridden method which satisfies polymorphism: R = -1
- If LOC does not occur any polymorphism: R = 0

S - Abstraction factor

- If a class is not an abstract class: S = 0
- If an abstract class or an interface: S = -1

En = Total weight of the n^{th} executable line of code

The above given values are generated by considering the increment or the decrement of the code complexity. Therefore, all the above-mentioned factor values should be added in order to get the En value for a particular line of code.

En = P + Q + R + S

Zn = Size of the n^{th} executable line of code in terms of the token count

a = Total number of executable code lines of a program

In order to calculate the complexity measure value of a particular code statement(line), we need to multiply the En value with the Zn value [En * Zn]. Therefore, to calculate the total complexity measure value of the whole program, we need to get the summation of each [En * Zn] value of each code line.

$$\boxed{\textbf{Object-Oriented Complexity Metric value of a program} = \sum_{n=1}^{a} E_n * Z_n}$$

Calculation of Code Complexity

Executable Java Program 1

```
public class Cake extends Foods implements greateFoods{
	public String additionalIngrediantsName = "icing";
	public int additionalIngrediantsPrice = 400;

	//polymorphism
	public void cut() {
		System.out.println("cake cut by smoothly");
	}

	@Override
	public void eats() {
		// TODO Auto-generated method stub
		System.out.println("eat flavoured foods");
	}

	@Override
	public void addflavours() {
		// TODO Auto-generated method stub
		System.out.println("add flavours");
	}
}

abstract public class Food {
	abstract void eat();

	void make() {
		System.out.println("Basic making mechanism of food\n");
	}
}

public class Foods {
	//encapsulation
	private String ingrediants;
	private int priceOfIngradiants;

	//inheritence
	public String additionalingrediantstype = "decoration";

	//inheritence
	public void addAditionals() {
		System.out.println("Adding additional ingrediants\n");
	}


	//polymorphism
	public void cut() {
		System.out.println("chicken cut by rough knife");
	}
```

```
      //encapsulation getters and setters

      /**
       * @return the ingrediants
       */
      public String getIngrediants() {
            return ingrediants;
      }
      /**
       * @param ingrediants the ingrediants to set
       */
      public void setIngrediants(String ingrediants) {
            this.ingrediants = ingrediants;
      }
      /**
       * @return the priceOfIngradiants
       */
      public int getPriceOfIngradiants() {
            return priceOfIngradiants;
      }
      /**
       * @param priceOfIngradiants the priceOfIngradiants to set
       */
      public void setPriceOfIngradiants(int priceOfIngradiants) {
            this.priceOfIngradiants = priceOfIngradiants;
      }

}

public interface greateFoods {
      public void eats();
      public void addflavours();
}

public class main {

      public static void main(String[] args) {
            // TODO Auto-generated method stub
            //abstraction code by using abstarct word
            Food chickenPizza = new pizza();

            chickenPizza.eat();
            chickenPizza.make();



            //abstraction by using interface

            //encapsulation
            Foods freshFoods = new Foods();

            freshFoods.setIngrediants("water + eggs");
            freshFoods.setPriceOfIngradiants(250);
```

```
                System.out.println("Ingrediants: " + freshFoods.getIngrediants());
                System.out.println("price of ingrediants: " +
freshFoods.getPriceOfIngradiants());
                System.out.println("\n");

                //inheritence
                Cake chocolatecake = new Cake();

                System.out.println("Additional Ingrediants name: " +
chocolatecake.additionalIngrediantsName);
                System.out.println("Additional Ingrediants price: " +
chocolatecake.additionalIngrediantsPrice);
                System.out.println("Additional Ingrediants type: " +
chocolatecake.additionalingrediantstype);

                chocolatecake.eats();
                chocolatecake.addflavours();

                chocolatecake.addAditionals();

                //polymorphism

                Foods chicken = new Foods();
                chicken.cut();

                Cake fruitcake = new Cake();
                fruitcake.cut();


        }

}

public class pizza extends Food{

        @Override
        void eat() {
                // TODO Auto-generated method stub
                System.out.println("Eat by cutting pieces");

        }

}
```

Total E = P + Q + R + S

Z(Size) = total number of tokens

Sum = E * Z

Program Statement	**Tokens**	**P** Encapsulation	**Q** Inheritance	**R** Polymorphism	**S** Abstraction	**Z** size	**E** Total	**E * Z** Sum	Explanation
`public class Cake extends Foods implements greateFoods{`									
`public String additionalIngrediantsName = "icing";`	String, additionalIngrediantsName, =, “icing”	1	3	0	0	4	4	16	Since there is no get or set method/ no encapsulation, P = 1. This class has extended a parent class and implemented an interface. Therefore, the inheritance is calculated with the equation Q = 1 + x + 1 : Q = 1 + 1 + 1 : Q = 3. There is no any overridden method in this LOC. So, R = 0. Since this LOC does not own to any abstract class or an interface S = 0. There are 4 tokens in the LOC, so the size Z = 4. Total value E = 4. Sum value = 4 X 4 = 16.
`public int additionalIngrediantsPrice = 400;`	int, additionalIngrediantsPrice, =, 400	1	3	0	0	4	4	16	No encapsulation: P = 1. Inheritance: Q = 1 + x + 1: Q = 3. No polymorphism: R = 0. No abstraction: S = 0 Z = 4 Sum = 4 X 4 = 16.
`public void cut() {`	void, cut()	1	3	0	0	2	4	8	No encapsulation: P = 1. Inheritance: Q = 1 + x + 1: Q = 3.

									No polymorphism: R = 0. No abstraction: S = 0 Z = 4 Sum = 2 X 4 = 8.
System.out.println("cake cut by smoothly");	System, . , out, . , println(),"cake cut by smoothly"	1	3	0	0	6	4	24	No encapsulation: P = 1. Inheritance: Q = 1 + x + 1: Q = 3. No polymorphism: R = 0. No abstraction: S = 0 Z = 6 Sum = 6 X 4 = 24.
}									
@Override public void eats() {	void, eats()	1	3	-1	0	2	3	6	No encapsulation: P = 1. Inheritance: Q = 1 + x + 1: Q = 3. eats() is an overridden method; therefore, polymorphism occurs: R = -1. No abstraction: S = 0 Z = 2 Sum = 2 X 3 = 6.
System.out.println("eat flavoured foods");	System, . , out, . , println(), "eat flavoured foods"	1	3	-1	0	6	3	18	No encapsulation: P = 1. Inheritance: Q = 1 + x + 1: Q = 3. Polymorphism occurs: R = -1. No abstraction: S = 0 Z = 6 Sum = 6 X 3 = 18.
}									
@Override public void addflavours() {	void, addflavours()	1	3	-1	0	2	3	6	No encapsulation: P = 1. Inheritance: Q = 1 + x + 1: Q = 3. addflavours() is an overridden method; therefore, polymorphism occurs: R = -1. No abstraction: S = 0

									Z = 2 Sum = 2 X 3 = 6.
`System.out.println("add flavours");`	System, . , out, . , println(), "add flavours"	1	3	-1	0	6	3	18	No encapsulation: P = 1. Inheritance: Q = 1 + x + 1: Q = 3. Polymorphism occurs: R = -1. No abstraction: S = 0 Z = 6 Sum = 6 X 3 = 18.
}									
}									
`abstract public class Food {`									
`abstract void eat();`	abstract, void, eat()	1	0	0	-1	3	0	0	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 Since class Food is an abstract class, abstraction occurs: S = -1 Z = 3 Sum = 0.
`void make() {`	void, make()	1	0	0	-1	2	0	0	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 Abstraction occurs: S = -1 Z = 2 Sum = 0.
`System.out.println("Basic making mechanism of food\n");`	System, . , out, . , println(), "Basic making mechanism of food\n"	1	0	0	-1	6	0	0	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 Abstraction occurs: S = -1 Z = 6 Sum = 0.
}									
}									

public class Foods {									
private String ingrediants;	String, ingrediants	1	0	0	0	2	1	2	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2 X 1 =2.
private int priceOfIngradiants;	int, priceOfIngradiants	1	0	0	0	2	1	2	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2 X 1 =2.
public String additionalingrediantstype = "decoration";	String, additionalingrediantstype , =, "decoration"	1	0	0	0	4	1	4	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = 1 X 4 = 4.
public void addAditionals() {	void, addAditionals()	1	0	0	0	2	1	2	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2 X 1 =2.
System.***out***.println("Adding additional ingrediants\n");	System, . , out, . , println(), "Adding additional ingrediants\n"	1	0	0	0	6	1	6	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 6 Sum = 6 X 1 =6.
}									
public void cut() {	void, cut()	1	0	0	0	2	1	2	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 2

									Sum = 2 X 1 =2.
System.out.println("chicken cut by rough knife");	System, . , out, . , println(), "chicken cut by rough knife"	1	0	0	0	6	1	6	No encapsulation: P = 1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 6 Sum = 6 X 1 = 6.
}									
public String getIngrediants() {	String, getIngredian ts()	-1	0	0	0	2	-1	-2	Encapsulation: P = -1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2 X -1 = -2.
return ingrediants;	return, ingredients	-1	0	0	0	2	-1	-2	Encapsulation: P = -1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2 X -1 = -2.
}									
public void setIngrediants(String ingrediants) {	void, setIngredian ts(), String, ingredients	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = 4 X -1 = -4.
this.ingrediants = ingrediants;	this, . , ingrediants, =, ingredients	-1	0	0	0	5	-1	-5	Encapsulation: P = -1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5 X -1 = -5.
}									
public int getPriceOfIngradiants() {	int, getPriceOfIn gradiants()	-1	0	0	0	2	-1	-2	Encapsulation: P = -1. Inheritance: Q = 0. No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2 X -1 = -2.

`return priceOfIngradiants;`	`return, priceOfIngradiants`	-1	0	0	0	2	-1	-2	Encapsulation: P = -1. Inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2 X -1 = -2.
`}`									
`public void setPriceOfIngradiants(int priceOfIngradiants) {`	`void, setPriceOfIngradiants(), int, priceOfIngradiants`	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = 4 X -1 = -4.
`this.priceOfIngradiants = priceOfIngradiants;`	`this, . , priceOfIngradiants, =, priceOfIngradiants`	-1	0	0	0	5	1	5	Encapsulation: P = -1. Inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5 X 1 = 5.
`}`									
`}`									
`public interface greateFoods {`									
`public void eats();`	`void, eats()`	1	0	0	-1	2	1	2	No encapsulation: P = 1. Inheritance: Q = 0. Polymorphism: R = 0 Since `greateFoods` is an interface abstraction occurs: S = -1 Z = 2 Sum = 2 X 1 = 2.
`public void addflavours();`	`void, addflavours()`	1	0	0	-1	2	1	2	No encapsulation: P = 1. Inheritance: Q = 0. Polymorphism: R = 0 Abstraction occurs: S = -1 Z = 2 Sum = 2 X 1 = 2.
`}`									
`public class main {`									

public static void main(String[] args) {	void, main(),String [], args	1	0	0	0	4	1	4	No encapsulation: P = 1. No inheritance since this is the main class: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = 4 X 1 = 4.
Food chickenPizza = new pizza();	Food, chickenPizza, =, new, pizza()	1	0	0	0	5	1	5	No encapsulation: P = 1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5 X 1 = 5.
chickenPizza.eat();	chickenPizza, . ,eat()	1	0	0	0	3	1	3	No encapsulation: P = 1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3 X 1 = 3.
chickenPizza.make();	chickenPizza, . ,make()	1	0	0	0	3	1	3	No encapsulation: P = 1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3 X 1 = 3.
Foods freshFoods = new Foods();	Foods, freshFoods, = ,new, Foods()	1	0	0	0	5	1	5	No encapsulation: P = 1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5 X 1 = 5.
freshFoods.setIngrediants("water + eggs");	freshFoods, . , setIngrediants (),"water + eggs"	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = -4

freshFoods.setPriceOfIngradiants(250);	freshFoods, . , setPriceOfIngradiants(), 250	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = -4
System.*out*.println("Ingrediants: " + freshFoods.getIngrediants());	System, . ,*out*, . ,println(),"Ingrediants: " , + , freshFoods, . ,getIngrediants()	-1	0	0	0	10	-1	-10	Encapsulation: P = -1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = -10
System.*out*.println("price of ingrediants: " + freshFoods.getPriceOfIngradiants());	System, . ,*out*, . ,println(),"price of ingrediants: ", +, freshFoods, . ,getPriceOfIngradiants()	-1	0	0	0	10	-1	-10	Encapsulation: P = -1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = -10
System.*out*.println("\n");	System, . ,*out*, . ,println() ,"\n"	1	0	0	0	6	1	6	No encapsulation: P = 1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 6 Sum = 6.
Cake chocolatecake = **new** Cake();	Cake, chocolatecake , =, **new,** Cake()	1	0	0	0	5	1	5	No encapsulation: P = 1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5 X 1 = 5.
System.*out*.println("Additional Ingrediants name: " + chocolatecake.additionalIngrediantsName);	System, . ,*out*, . ,println(),"Additional Ingrediants name: ", +, chocolatecake , . ,additionalIngrediantsName	1	0	0	0	10	1	10	No encapsulation: P = 1. No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = 10.
System.*out*.println("Additional Ingrediants	System, . ,*out*, . ,println(),"A	1	0	0	0	10	1	10	No encapsulation: P = 1.

price: " + chocolatecake.additionalIngrediantsPrice);	dditional Ingrediants price: ", +, chocolatecake , . ,additionalIngrediantsPrice								No inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = 10.
System.*out*.println("Additional Ingrediants type: " + chocolatecake.additionalingrediantstype);	System, . ,*out*, . ,println(),"Additional Ingrediants type: " , + , chocolatecake , . ,additionalingrediantstype	1	0	0	0	10	1	10	No encapsulation: P = 1. No inheritance: Q = 0. No polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = 10.
chocolatecake.eats();	chocolatecake , . ,eats()	1	0	-1	0	3	1	3	No encapsulation: P = 1. No inheritance: Q = 0 Polymorphism used with the eats() overridden function, so R = -1. No abstraction: S = 0
chocolatecake.addflavours();	chocolatecake , . ,addflavours()	1	0	-1	0	3	1	3	No encapsulation: P = 1. No inheritance: Q = 0 Polymorphism used with the addflavours() overridden function, so R = -1. No abstraction: S = 0
chocolatecake.addAditionals();	Chocolatecake , . ,addAditionals()	1	0	0	0	3	1	3	No encapsulation: P = 1. Inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3.
Foods chicken = new Foods();	Foods, chicken, =, new, Foods()	1	0	0	0	5	1	5	No encapsulation: P = 1. Inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5.
chicken.cut();	chicken, . ,cut()	1	0	0	0	3	1	3	No encapsulation: P = 1.

									Inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3.
`Cake fruitcake = new Cake();`	Cake, fruitcake, =, new, Cake()	1	0	0	0	5	1	5	No encapsulation: P = 1. Inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5.
`fruitcake.cut();`	fruitcake, . ,cut()	1	0	0	0	3	1	3	No encapsulation: P = 1. Inheritance: Q = 0. Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3
}									
}									
`public class pizza extends Food{`									
`void eat() {`	void, eat()	1	1	0	0	2	1	2	No encapsulation: P = 1. Inheritance: Q = 1. Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2
`System.out.println("Eat by cutting pieces");`	System, . ,out, . , println(),"Eat by cutting pieces"	1	1	0	0	6	1	6	No encapsulation: P = 1. Inheritance: Q = 1. Polymorphism: R = 0 No abstraction: S = 0 Z = 6 Sum = 6
}									
}									
WCC								**190**	

Executable Java Program 2

```java
public class Car extends Vehicle{

        @Override
        void drive() {
                // TODO Auto-generated method stub
                System.out.println("drive slowly\n");
        }

}

public interface classicferrari {
        public void accelaration();
        public void suspensions();
}


public class ferrari extends ModernVehicles implements moderferrari, classicferrari{
        public String ferrariname = "spider";

        @Override
        public void accelaration() {
                // TODO Auto-generated method stub
                System.out.println("have best accelaration based on consumption");
        }

        @Override
        public void suspensions() {
                // TODO Auto-generated method stub
                System.out.println("hydrolic suspension");
        }

        @Override
        public void speed() {
                // TODO Auto-generated method stub
                System.out.println("Top speed 320kmh");
        }

        @Override
        public void fuelconsumption() {
                // TODO Auto-generated method stub
                System.out.println("11.4/liters");
        }
}

import java.io.ObjectInputStream.GetField;

public class main {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
        //encapsulation
        ModernVehicles brandedVehicle = new ModernVehicles();

        brandedVehicle.setVehicleBrand("Nissan");

        System.out.println("Vehicle Brand: " +
brandedVehicle.getVehicleBrand());
        System.out.println("Vehicle name: " + brandedVehicle.vehicleName);
        System.out.println("\n");


        //abstraction
        Vehicle audiCar = new Car();

        audiCar.drive();


        //inheritence
        ferrari  newferraispider = new ferrari();

        System.out.println("BodyKit name" + newferraispider.bodykit);
        System.out.println("Ferrai name" + newferraispider.ferrariname);
        newferraispider.accelaration();
        newferraispider.bodykitPosition();
        newferraispider.fuelconsumption();
        newferraispider.speed();
        newferraispider.suspensions();
        System.out.println();


        //polymorphism

        ModernVehicles basicvehiclelogo = new ModernVehicles();
        basicvehiclelogo.logo();

        ModernVehicles lamborghini = new lamborghini();
        lamborghini.logo();



    }

}


public interface moderferrari {
    public void speed();
    public void fuelconsumption();
}


public class lamborghini extends ModernVehicles{
    public void logo() {
        System.out.println("lamborghini logo");
    }
```

```java
}


public class ModernVehicles {
	private String vehicleBrand;
	public String vehicleName = "GTR";

	//inheritence
	public String bodykit = "full";

	//inheritence
	public void bodykitPosition() {
		System.out.println("set bodykit position");
	}


	//polymorphism
	public void logo() {
		System.out.println("basic logo for vehicles");
	}

	/**
	 * @return the vehicleBrand
	 */
	public String getVehicleBrand() {
		return vehicleBrand;
	}
	/**
	 * @param vehicleBrand the vehicleBrand to set
	 */
	public void setVehicleBrand(String vehicleBrand) {
		this.vehicleBrand = vehicleBrand;
	}



}

abstract public class Vehicle {
	abstract void drive();

}
```

Program Statement	**Tokens**	**P** Encapsulation	**Q** Inheritance	**R** Polymorphism	**S** Abstraction	**Z** size	**E** Total	**E * Z** Sum	Explanation
`public class Car extends Vehicle{`									

`@Override`
`void drive() {`
	`void, drive()`	1	1	-1	0	2	1	2	There is no getter or setter method in this LOC; so, no encapsulation: P = 1.
									Since the Car class is inherited from Vehicle class, Inheritance: Q = 1
									The drive() method is an overridden method; therefore, Polymorphism: R = -1
									Since this is not an abstract class or an interface, no abstraction occurs: S = 0
									Z = 2
									Sum = 2.

`System.out.println("drive slowly\n");`	`System, . , out, . ,println(), "drive slowly\n”`	1	1	-1	0	6	1	6	No encapsulation: P = 1. Inheritance: Q = 1 Polymorphism occurs: R = -1 No abstraction: S = 0 Z = 6 Sum = 6.
`}`									
`}`									
`public interface classicferrari {`									
`public void accelaration();`	`void, accelaration()`	1	0	0	-1	2	0	0	No encapsulation: P = 1. No inheritance since this is an interface: Q = 0 No

									Polymorphism: R = 0 Since this is an interface abstraction occurs: S = -1 Z = 0 Sum = 0.
`public void suspensions();`	`void, suspensions()`	1	0	0	-1	2	0	0	No encapsulation: P = 1. No inheritance since this is an interface: Q = 0 No Polymorphism: R = 0 Abstraction occurs: S = -1 Z = 0 Sum = 0.
`}`									
`public class ferrari extends ModernVehicles implements moderferrari, classicferrari{`									
`public String ferrariname = "spider";`	`String, ferrariname =, "spider"`	1	4	0	0	4	5	20	There is no getter or setter method in this LOC; so, no encapsulation: P = 1. The class Ferrari has extended from a parent class and has implemented two interfaces. Therefore Q value should be calculated as Q = 1+x+1: Q = 1+2+1: Q = 4. No overridden method; so Polymorphism: R = 0 No abstraction: S = 0 Z = 4

									Sum = 20.
@Override public void accelaration() {	void, accelaration( )	1	4	-1	0	2	4	8	No encapsulation: P = 1. Inheritance: Q = 4 celaration() is an overridden method. Therefore Polymorphism: R = -1 No abstraction: S = 0 Z = 4 Sum = 8.
System.out.println("have best accelaration based on consumption");	System, . ,out, . ,println(), "have best accelaration based on consumption"	1	4	-1	0	6	4	24	No encapsulation: P = 1. Inheritance: Q = 4 Polymorphism occurs: R = -1 No abstraction: S = 0 Z = 6 Sum = 24.
}									
@Override public void suspensions() {	void ,suspensions( )	1	4	-1	0	2	4	8	No encapsulation: P = 1. Inheritance: Q = 4 Polymorphism: R = -1 No abstraction: S = 0 Z = 2 Sum = 8.
System.out.println("hydrolic suspension");	System, . ,out, ., println(), "hydrolic suspension"	1	4	-1	0	6	4	24	No encapsulation: P = 1. Inheritance: Q = 4 Polymorphism: R = -1 No abstraction: S = 0 Z = 6 Sum = 24.
}									
@Override public void speed() {	void, speed()	1	4	-1	0	2	4	8	No encapsulation: P = 1.

									Inheritance: Q = 4 Polymorphism: R = -1 No abstraction: S = 0 Z = 2 Sum = 8.
System.out.println("Top speed 320kmh");	System, ., out, ., println(), "Top speed 320kmh"	1	4	-1	0	6	4	24	No encapsulation: P = 1. Inheritance: Q = 4 Polymorphism: R = -1 No abstraction: S = 0 Z = 6 Sum = 24.
}									
@Override public void fuelconsumption() {	void , fuelconsumption()	1	4	-1	0	2	4	8	No encapsulation: P = 1. Inheritance: Q = 4 Polymorphism: R = -1 No abstraction: S = 0 Z = 2 Sum = 8.
System.out.println("11.4/liters");	System, . , out, ., println(), "11.4/liters"	1	4	-1	0	6	4	24	No encapsulation: P = 1. Inheritance: Q = 4 Polymorphism: R = -1 No abstraction: S = 0 Z = 6 Sum = 24.
}									
}									
public class lamborghini extends ModernVehicles{									
public void logo() {	void ,logo()	1	1	0	0	2	2	4	No encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0

									No abstraction: S = 0 Z = 2 Sum = 4.
System.out.println("lamborghini logo");	System, . ,out, . , println(),"lamborghini logo"	1	1	0	0	6	2	12	No encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No abstraction: S = 0 Z = 6 Sum = 12
}									
}									
import java.io.ObjectInputStream.GetField;									
public class main {									
public static void main(String[] args) {	static ,void, main(), String[], args	1	0	0	0	5	1	5	No encapsulation: P = 1. No inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5
ModernVehicles brandedVehicle = new ModernVehicles();	ModernVehicles , brandedVehicle , = new , ModernVehicles()	1	0	0	0	5	1	5	No encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5
brandedVehicle.setVehicleBrand("Nissan");	brandedVehicle , . , setVehicleBrand() ,"Nissan"	-1	0	0	0	4	-1	-4	Since this line contains a setter method, encapsulation occurs: P = -1.

									Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = -4
System.out.println("Vehicle Brand: " + brandedVehicle.getVehicleBrand());	System,. , out, . ,println(),"Vehicle Brand: " , + , brandedVehicle, . , getVehicleBrand()	-1	0	0	0	10	-1	-10	Encapsulation occurs: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = -10
System.out.println("Vehicle name: " + brandedVehicle.vehicleName);	System, . , out, ., println(),"Vehicle name: " , + , brandedVehicle, . , vehicleName	1	0	0	0	10	1	10	No Encapsulation: P = 1 Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = -10
System.out.println("\n");	System , . , out, . , println(), "\n"	1	0	0	0	6	1	6	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 6 Sum = 6
Vehicle audiCar = new Car();	Vehicle, audiCar, = , new , Car()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0

									Z = 5 Sum = 5
audiCar.drive();	audiCar , . , drive()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3
ferrari newferraispider = new ferrari();	ferrari , newferraispider , = , new , ferrari()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5
System.out.println("BodyKit name" + newferraispider.bodykit);	System, . , out, . , println(), "BodyKit name", + , newferraispider, . , bodykit()	1	0	0	0	10	1	10	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = 10
System.out.println("Ferrai name" + newferraispider.ferrariname);	System, . ,out, . ,println(), "Ferrai name" , + , newferraispider, . ,ferrariname	1	0	0	0	10	1	10	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 10 Sum = 10
newferraispider.accelaration();	Newferraispider, . ,accelaration ()	1	0	-1	0	3	0	0	No Encapsulation: P = 1.

									Inheritance: Q = 0 This LOC contains the overridden method `accelaration().` Therefore, polymorphism occurs: R = -1 No abstraction: S = 0 Z = 3 Sum = 0
`newferraispider.bodykitPosition();`	`Newferraispider, . ,bodykitPosition()`	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3
`newferraispider.fuelconsumption();`	`Newferraispider , . ,fuelconsumption()`	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 This LOC contains the overridden method `fuelconsumptio().` Therefore, polymorphism occurs: R = -1 No abstraction: S = 0 Z = 3 Sum = 0
`newferraispider.speed();`	`Newferraispider, . , speed()`	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 This LOC contains the overridden method `speed().` Therefore, polymorphism occurs: R = -1 No abstraction: S = 0

									Z = 3 Sum = 0
newferraispider.suspensions();	Newferraispider, . , suspensions()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 This LOC contains the overridden method suspensions(). Therefore, polymorphism occurs: R = -1 No abstraction: S = 0 Z = 3 Sum = 0
System.out.println();	System, . ,out , . , println()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5
ModernVehicles basicvehiclelogo = new ModernVehicles();	ModernVehicles , basicvehiclelogo, = , new , ModernVehicles()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5
basicvehiclelogo.logo();	basicvehiclelogo, . ,logo()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3

ModernVehicles lamborghini = new lamborghini();	ModernVehicles , Lamborghini, = ,new , lamborghini()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = 5
lamborghini.logo();	lamborghini, . , logo()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 3 Sum = 3
}									
}									
public interface moderferrari {									
public void speed();	void , speed()	1	0	0	-1	2	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 Since moderferrari is an interface abstraction occurs: S = -1 Z = 2 Sum = 0
public void fuelconsumption();	void ,fuelconsumption()	1	0	0	-1	2	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 Abstraction occurs: S = -1 Z = 2 Sum = 0

}									
public class ModernVehicles {									
private String vehicleBrand;	String , vehicleBrand	1	0	0	0	2	1	2	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2
public String vehicleName = "GTR";	String, vehicleName , = , "GTR"	1	0	0	0	4	1	4	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = 4
public String bodykit = "full";	String , bodykit, = , "full"	1	0	0	0	4	1	4	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = 4
public void bodykitPosition() {	void ,bodykitPosition()	1	0	0	0	2	1	2	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2
System.*out*.println("set bodykit position");	System , . ,*out*, . , println(),"se	1	0	0	0	6	1	6	No Encapsulation: P = 1. Inheritance: Q = 0 No

	t bodykit position"								Polymorphism: R = 0 No abstraction: S = 0 Z = 6 Sum = 6
}									
public void logo() {	void ,logo()	1	0	0	0	2	1	2	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = 2
System.out.println("basic logo for vehicles");	System, . , out, . , println(),"basic logo for vehicles"	1	0	0	0	6	1	6	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 6 Sum = 6
}									
public String getVehicleBrand() {	String ,getVehicleBrand()	-1	0	0	0	2	-1	-2	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 2 Sum = -2
return vehicleBrand;	return ,vehicleBrand	-1	0	0	0	2	-1	-2	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0

									Z = 2 Sum = -2
}									
`public void setVehicleBrand(String vehicleBrand) {`	`void , setVehicleBrand(), String ,vehicleBrand`	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 4 Sum = -4
`this.vehicleBrand = vehicleBrand;`	`this , . , vehicleBrand , = , vehicleBrand`	-1	0	0	0	5	-1	-5	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No abstraction: S = 0 Z = 5 Sum = -5
}									
}									
`abstract public class Vehicle {`									
`abstract void drive();`	`void ,drive()`	1	0	0	-1	2	1	2	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 Abstraction: S = -1 Z = 2 Sum = 2
}									
WCC								**256**	

Executable Java Program 3

```java
public interface Employee {

        public void makeMeeting();
        public void promoted();
}


public class main {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                //abstraction using interfaces
                Manager newmanager = new Manager();

                newmanager.makeMeeting();
                newmanager.promoted();
                System.out.println();

                //encapsulation
                Manager oldmanager = new Manager();

                oldmanager.setSalary(90000.00);
                oldmanager.setPosition("senior");

                System.out.println("Position: " + oldmanager.getPosition());
                System.out.println("Salary: " + oldmanager.getSalary());
                System.out.println();

                //inheritance
                Managingdirector md = new Managingdirector();
                System.out.println("EPF: " + md.EPF);
                System.out.println("Bonus : " + md.bonus);
                System.out.println();

                md.interviewing();

                //polymorphism
                Manager manager = new Manager();
                manager.transportation();

                Manager managingdir = new Managingdirector();
                managingdir.transportation();

        }

}


public class Manager implements Employee{

        private double salary;
```

```java
    private String position;

    //inheritance
    public double EPF = 78000.00;

    public void interviewing() {
        System.out.println("Interview other employees");
    }

    //polymorphism
    public void transportation() {
        System.out.println("come by car");
    }



    //abstraction
    @Override
    public void makeMeeting() {
        // TODO Auto-generated method stub
        System.out.println("make new meeting with other employe");
    }

    @Override
    public void promoted() {
        // TODO Auto-generated method stub
        System.out.println("get promoted to new position");
    }

    //encapsulation
    /**
     * @return the salary
     */
    public double getSalary() {
        return salary;
    }

    /**
     * @param salary the salary to set
     */
    public void setSalary(double salary) {
        this.salary = salary;
    }

    /**
     * @return the position
     */
    public String getPosition() {
        return position;
    }

    /**
     * @param position the position to set
     */
    public void setPosition(String position) {
```

```
                this.position = position;
        }



}

public class Managingdirector extends Manager{
        public double bonus = 56000.00;

        public void transportation() {
                System.out.println("come by luxuary car");
        }
}
```

E(Total) = We + Wi + Wp + Wa

Sum = E * Z

Program Statement	**Tokens**	**P** Encapsulation **(We)**	**Q** Inheritance **(Wi)**	**R** Polymorphism **(Wp)**	**S** Abstraction **(Wa)**	**Z** size	**E** Total	**E * Z** Sum	**Explanation**
`public interface Employee {`									
`public void makeMeeting();`	`void, makeMeeting()`	1	0	0	-1	2	0	0	There is no getter or setter method; so, no Encapsulation: P = 1. This interface has not extended; so, no inheritance: Q = 0. This LOC has no overridden method; therefore, no polymorphism occurs: R = 0. Since this is an interface abstraction occurs: S = -1 Z = 2 Sum = 0
`public void promo0ted();`	`void , promoted()`	1	0	0	-1	2	0	0	No Encapsulation: P = 1.

									Inheritance: Q = 0 No Polymorphism: R = 0 Abstraction occurs: S = -1 Z = 2 Sum = 0
}									
public class main {									
public static void main(String[] args) {	Static, void, main(), String[] args	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
Manager newmanager = new Manager();	Manager , newmanager , = ,new Manager()	1	0	0	0	4	1	4	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 4
newmanager.makeMeeting();	newmanager, . ,makeMeeting()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Since this LOC contains an overridden method polymorphism occurs: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
newmanager.promoted();	Newmanager , . , promoted()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Polymorphism: R = -1 No Abstraction: S = 0

									Z = 3 Sum = 0
System.out.println();	System, . , out, . , println()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
Manager oldmanager = new Manager();	Manager , oldmanager , = , new , Manager()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
oldmanager.setSalary(90000.00);	Oldmanager, . ,setSalary() ,90000.00	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = -4
oldmanager.setPosition("senior");	Oldmanager , . , setPosition() ,"senior"	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = -4
System.out.println("Position: " + oldmanager.getPosition());	System, . , out, . , println(),"Position: " , + , oldmanager, . , getPosition()	-1	0	0	0	10	-1	-10	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 10

									Sum = -10
`System.out.println("Salary: " + oldmanager.getSalary());`	System, . , out, . , println(), "Salary: ", + , oldmanager , . , getSalary()	-1	0	0	0	10	-1	-10	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 10 Sum = -10
`System.out.println();`	System, . , out, . , println()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
`Managingdirector md = new Managingdirector();`	Managingdirector , md , = , new ,Managingdirector()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
`System.out.println("EPF: " + md.EPF);`	System, . ,out , . ,println(), "EPF: " , + , md , . , EPF	1	0	0	0	10	1	10	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 10 Sum = 10
`System.out.println("Bonus : " + md.bonus);`	System, . , out, . , println(), "Bonus : " , + , md , . ,bonus	1	0	0	0	10	1	10	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 10 Sum = 10

System.out.println();	System, . ,out ,. , println()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
md.interviewing();	md, . ,interviewing()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 3
Manager manager = new Manager();	Manager , manager , = , new , Manager()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
manager.transportation();	Manager, . , transportation()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 3
Manager managingdir = new Managingdirector();	Manager , managingdir , = , new , Managingdirector()	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5

managingdir.transportation();	Managingdir, . , transportation()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 3
}									
}									
public class Manager implements Employee{									
private double salary;	double , salary	1	2	0	0	2	3	6	No Encapsulation: P = 1. Since this class has implemented an interface inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
private String position;	String , position	1	2	0	0	2	3	6	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
public double EPF = 78000.00;	double ,EPF, = ,78000.00	1	2	0	0	4	3	12	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 12

public void interviewing() {	void , interviewing()	1	2	0	0	2	3	6	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
System.out.println("Interview other employees");	System, . ,out, . , println(), "Interview other employees"	1	2	0	0	6	3	18	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 18
}									
public void transportation() {	void, transportation()	1	2	0	0	2	3	6	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
System.out.println("come by car");	System, . , out, . , println() ,"come by car"	1	2	0	0	6	3	18	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 18
}									
@Override public void makeMeeting() {	void , makeMeeting()	1	2	-1	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 2 Since this is an overridden method, Polymorphism: R = -1 No Abstraction: S = 0

									Z = 2 Sum = 4
System.out.println("make new meeting with other employe");	System, . , out, . ,println() ,"make new meeting with other employe"	1	2	-1	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 6 Sum = 12
}									
@Override public void promoted() {	void ,promoted()	1	2	-1	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 4
System.out.println("get promoted to new position");	System, . , out, . , println(),"get promoted to new position"	1	2	-1	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 12
}									
public double getSalary() {	double, getSalary()	-1	2	0	0	2	1	2	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 2
return salary;	return ,salary	-1	2	0	0	2	1	2	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0

									No Abstraction: S = 0 Z = 2 Sum = 2
}									
public void setSalary(double salary) {	void, setSalary(),double, salary	-1	2	0	0	4	1	4	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 4
this.salary = salary;	this, . , salary , = , salary	-1	2	0	0	5	1	5	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
}									
public String getPosition() {	String, getPosition()	-1	2	0	0	2	1	2	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 2
return position;	return , position	-1	2	0	0	2	1	2	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 2
}									
public void setPosition(String position) {	void ,setPosition(	-1	2	0	0	4	1	4	Encapsulation: P = -1.

	),String position								Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 4
this.position = position;	his , . , position , = position	-1	2	0	0	5	1	5	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
}									
}									
public class Managingdirector extends Manager{									
public double bonus = 56000.00;	public, double, bonus, = , 56000.00;	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
public void transportation() {	public, void ,transportation()	1	1	0	0	3	2	6	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 6
System.out.println("come by luxuary car");	System, . , out, . ,println(), "come by luxuary car"	1	1	0	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6

									Sum = 12
}									
}									
WCC								203	

Executable Java Program 4

```
public class SmartPhone extends Phone {

        private String ringingTone;
        private String operatingSystem;
        private String screenSize;
        private String batteryCapacity;
        private boolean messageReceived;
        private String message;
        private String brand;
        private String model;

        public SmartPhone(int number) {
                super(number);
        }

        public SmartPhone(int number, String ringingTone, String operatingSystem,
String screenSize, String batteryCapacity,
                        String brand, String model) {
                super(number);
                this.ringingTone = ringingTone;
                this.operatingSystem = operatingSystem;
                this.screenSize = screenSize;
                this.batteryCapacity = batteryCapacity;
                this.brand = brand;
                this.model = model;
        }

        public String getRingingTone() {
                return ringingTone;
        }

        public void setRingingTone(String ringingTone) {
                this.ringingTone = ringingTone;
        }

        public String getOperatingSystem() {
                return operatingSystem;
        }
```

```
    public void setOperatingSystem(String operatingSystem) {
        this.operatingSystem = operatingSystem;
    }

    public String getScreenSize() {
        return screenSize;
    }

    public void setScreenSize(String screenSize) {
        this.screenSize = screenSize;
    }

    public String getBatteryCapacity() {
        return batteryCapacity;
    }

    public void setBatteryCapacity(String batteryCapacity) {
        this.batteryCapacity = batteryCapacity;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    @Override
    public void details() {
        System.out.println("This is a " + brand + " " + model + " smart phone");
    }

    @Override
    public void isRinging() {
        System.out.println("Smart phone is ringing with" + this.ringingTone +
"ringing tone");
    }

    public void sendMessage(int number, String message) {
        if (number == this.number) {
            this.message = message;
            messageReceived = true;
        }
    }

    public void isMessaging() {
```

```java
            if (messageReceived) {
                System.out.println("New message received");
            }
        }

        public void readMessage() {
            System.out.println("Message: " + this.message);
        }

    }


    public class Phone implements Device {

        protected int number;
        protected boolean ringing;

        public Phone(int number) {

            this.number = number;
        }

        public int getNumber() {
            return number;
        }

        public void setNumber(int number) {
            this.number = number;
        }

        @Override
        public void switchOn() {
            System.out.println("Phone switched on");

        }

        @Override
        public void switchOff() {
            System.out.println("Phone switched off");

        }

        public void isRinging() {
            if (ringing) {
                System.out.println("Phone is ringing");
            }
        }

        public void call(int number) {
            if (number == this.number) {
                this.ringing = true;
            }

        }
        public void answer() {
            if (ringing) {
                System.out.println("Answering the call");
                ringing = false;
            }
        }

        @Override
        public void details() {
            System.out.println("This is a phone");
        }

    }


    public class LandlinePhone extends Phone {
        private int numberOfKeys;
        private boolean cliFeature;
        private String brand;
        private String model;

        public LandlinePhone(int number) {
            super(number);

        }

        public LandlinePhone(int number, int numberOfKeys, boolean cliFeature, String 
    brand, String model) {
            super(number);
            this.numberOfKeys = numberOfKeys;
            this.cliFeature = cliFeature;
            this.brand = brand;
            this.model = model;
        }

        public int getNumberOfKeys() {
            return numberOfKeys;
        }

        public void setNumberofKeys(int numberOfKeys) {
            this.numberOfKeys = numberOfKeys;
        }

        public boolean isCliFeature() {
            return cliFeature;
        }

        public void setCliFeature(boolean cliFeature) {
            this.cliFeature = cliFeature;
        }

        public String getBrand() {
            return brand;
        }
        public void setBrand(String brand) {
            this.brand = brand;
        }

        public String getModel() {
            return model;
        }

        public void setModel(String model) {
            this.model = model;
        }

        @Override
        public void isRinging() {
            if (ringing) {
                System.out.println("Lanline phone is ringing");
            }
        }

        @Override
        public void details() {
            System.out.println("This is a " + brand + " " + model + " Landline
    Phone");
        }

    }


    public interface Device {
        public void switchOn();

        public void switchOff();

        public void details();

    }



    public class Main {

        public static void main(String[] args) {
            //Runtime polymorphism
            Device device1 = new Phone("0112845678");
            device1.details();


            Phone phone1 = new Phone("0778902245");
            phone1.details();


            Phone phone2 = new SmartPhone("0718900817","Nokia ", "Android 8.0",
    "5.5", "3000 mAh","Nokia", "6");
            //Runtime polymorphism
            phone2.details();
```

```
                phone2.call("0718900817");
                phone2.isRinging();
                phone2.answer();


                SmartPhone smartPhone = new 
SmartPhone("0778902345","Reflection","ios","5.8","2700mAh","Apple","iPhone x");
                smartPhone.details();
                smartPhone.call("0778902345");
                smartPhone.isRinging();
                smartPhone.answer();
                smartPhone.sendMessage("0778902345","hiii");
                smartPhone.isMessaging();
                smartPhone.readMessage();

                //Compile time polymorphism --->See the usage of overloaded constructors
                LandlinePhone landLinePhone = new LandlinePhone("01126456789");
                landLinePhone.setBrand("Tp-Link");
                landLinePhone.setModel("t6");
                landLinePhone.setCliFeature(false);
                landLinePhone.setNumberofKeys(20);

                landLinePhone.details();

                //Compile time polymorphism --->See the usage of overloaded constructors
                LandlinePhone landLinePhone2 = new 
LandlinePhone("01126456789",21,true,"Panasonic","R6");
                landLinePhone.details();
                landLinePhone2.call("01126456789");
                landLinePhone2.isRinging();
                landLinePhone2.answer();



        }

}
```

Program Statement	**Tokens**	**P** Encapsulation	**Q** Inheritance	**R** Polymorphism	**S** Abstraction	**Z** size	**E** Total	**E** ***** **Z** Sum	Explanation
`public class SmartPhone extends Phone {`									
`private String ringingTone;`	String, `ringingTone`	1	1	0	0	2	2	4	There are no getter or setter methods, no encapsulation: P = 1. Class has extended from a parent class, inheritance: Q = 1

									No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private String operatingSystem;	String, operatingSystem	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private String screenSize;	String, screenSize	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private String batteryCapacity;	String, batteryCapacity	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private boolean messageReceived;	boolean, messageReceived	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private String message;	String, message	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private String brand;	String, brand	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0

									No Abstraction: S = 0 Z = 2 Sum = 4
private String model;	String, model	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
public SmartPhone(int number) {	SmartPhone() , int, number	1	1	0	0	3	2	6	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 6
super(number);	Super(), number	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
}									
public SmartPhone(int number, String ringingTone, String operatingSystem, String screenSize, String batteryCapacity,String brand, String model) {	SmartPhone() , int, number, String, ringingTone, String, operatingSystem , String, screenSize, String, batteryCapacity, String, brand, String, model	1	1	0	0	15	2	30	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 15 Sum = 30
super(number);	Super(), number	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2

									Sum = 4
this.ringingTone = ringingTone;	this,., ringingTone, =, ringingTone	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
this.operatingSystem = operatingSystem;	this, ., operatingSystem, =, operatingSystem	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
this.screenSize = screenSize;	this, ., screenSize, =, screenSize	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
this.batteryCapacity = batteryCapacity;	this, ., batteryCapacity, =, batteryCapacity	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
this.brand = brand;	this, ., brand, =, brand	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
this.model = model;	this, ., model, =, model	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
}									

public String getRingingTone() {	String, getRingingTone()	-1	1	0	0	2	0	0	Contains a get method, encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
return ringingTone;	Return, ringingTone	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
}									
public void setRingingTone(String ringingTone) {	Void, setRingingTone(), String, ringingTone	-1	1	0	0	4	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 0
this.ringingTone = ringingTone;	This, . , ringingTone, =, ringingTone	-1	1	0	0	5	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 0
}									
public String getOperatingSystem() {	String, getOperatingSystem()	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
return operatingSystem;	return, operatingSystem	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
}									
public void setOperatingSystem(String operatingSystem) {	void, setOperatingSystem(), String,	-1	1	0	0	4	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0

	operatingSystem								No Abstraction: S = 0 Z = 4 Sum = 0
this.operatingSystem = operatingSystem;	this, . , operatingSystem, =, operatingSystem	-1	1	0	0	5	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 0
}									
public String getScreenSize() {	String, getScreenSize()	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
return screenSize;	return, screenSize	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
}									
public void setScreenSize(String screenSize) {	Void, setScreenSize(), String, screenSize	-1	1	0	0	4	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 0
this.screenSize = screenSize;	this, ., screenSize, =, screenSize	-1	1	0	0	5	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 0
}									
public String getBatteryCapacity() {	String, getBatteryCapacity()	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0

`return batteryCapacity;`	`return, batteryCapacity`	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
`}`									
`public void setBatteryCapacity(String batteryCapacity) {`	`void, setBatteryCapacity(), String, batteryCapacity`	-1	1	0	0	4	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 0
`this.batteryCapacity = batteryCapacity;`	`This, . , batteryCapacity, =, batteryCapacity`	-1	1	0	0	5	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 0
`}`									
`public String getBrand() {`	`String, getBrand()`	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
`return brand;`	`return, brand`	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
`}`									
`public void setBrand(String brand) {`	`Void, setBrand(), String, brand`	-1	1	0	0	4	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 0
`this.brand = brand;`	`This, . , brand, =, brand`	-1	1	0	0	5	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0

									Z = 5 Sum = 0
}									
public String getModel() {	String, getModel()	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
return model;	return, model	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
}									
public void setModel(String model) {	void, setModel(), String, model	-1	1	0	0	4	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 0
this.model = model;	this, . , model, =, model	-1	1	0	0	5	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 0
}									
@Override public void details() {	void, details()	1	1	-1	0	2	1	2	No Encapsulation: P = 1. Inheritance: Q = 1 Contains an overridden method, Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 2
System.out.println ("This is a " + brand + " " + model + " smart phone");	System, ., out, ., println(), "This is a ", +, brand, +, " ", +, model, +, " smart phone"	1	1	-1	0	14	1	14	No Encapsulation: P = 1. Inheritance: Q = 1 Polymorphism: R = -1 No Abstraction: S = 0 Z = 14 Sum = 14

}									
@Override public void isRinging() {	void, isRinging()	1	1	-1	0	2	1	2	No Encapsulation: P = 1. Inheritance: Q = 1 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 2
System.out.println ("Smart phone is ringing with" + this.ringingTone + "ringing tone");	System, ., out, ., println(), "Smart phone is ringing with", +, this, ., ringingTone, +, "ringing tone"	1	1	-1	0	12	1	12	No Encapsulation: P = 1. Inheritance: Q = 1 Polymorphism: R = -1 No Abstraction: S = 0 Z = 12 Sum = 12
}									
public void sendMessage(int number, String message) {	void, sendMessage(), int, number, String, message	1	1	0	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 12
if (number == this.number) {	if(), number, ==, this, ., number	1	1	0	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 12
this.message = message;	this, ., message, =, message	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
messageReceived = true;	messageReceived, =, true	1	1	0	0	3	2	6	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 6

}									
}									
public void isMessaging() {	void, isMessaging()	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
if (messageReceived) {	if(), messageReceived	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
System.out.println ("New message received");	System, ., out, ., println(), "New message received"	1	1	0	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 12
}									
}									
public void readMessage() {	void, readMessage()	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
System.out.println ("Message: " + this.message);	System, ., out, ., println(), "Message: ", +, this, ., message	1	1	0	0	10	2	20	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 10 Sum = 20
}									
}									
public class Phone implements Device {									

`protected int number;`	`int, number`	1	2	0	0	2	3	6	No Encapsulation: P = 1. Has implemented an interface, Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
`protected boolean ringing;`	`boolean, ringing`	1	2	0	0	2	3	6	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
`public Phone(int number) {`	`Phone(), int, number`	1	2	0	0	3	3	9	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 9
`this.number = number;`	`this, ., number, =, number`	1	2	0	0	5	3	15	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 15
`}`									
`public int getNumber() {`	`int, getNumber()`	-1	2	0	0	2	1	2	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 2
`return number;`	`return, number`	-1	2	0	0	2	1	2	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 2
`}`									

public void setNumber(int number) {	void, setNumber(), int, number	-1	2	0	0	4	1	4	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 4
this.number = number;	this, ., number, =, number	-1	2	0	0	5	1	5	Encapsulation: P = -1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 5
}									
@Override public void switchOn() {	void, switchOn()	1	2	-1	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 4
System.out.println ("Phone switched on");	System, ., out, ., println(), "Phone switched on"	1	2	-1	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 6 Sum = 12
}									
@Override public void switchOff() {	void, switchOff()	1	2	-1	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 4
System.out.println ("Phone switched off");	System, ., out, ., println(), "Phone switched off"	1	2	-1	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 6 Sum = 12
}									
public void isRinging() {	void, isRinging()	1	2	0	0	2	3	6	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0

									No Abstraction: S = 0 Z = 2 Sum = 6
if (ringing) {	if(), ringing	1	2	0	0	2	3	6	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
System.*out*.println ("Phone is ringing");	System, ., *out*, ., println(), "Phone is ringing"	1	2	0	0	6	3	18	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 18
}									
}									
public void call(int number) {	void, call(), int, number	1	2	0	0	4	3	12	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 12
if (number == this.number) {	if(), number, ==, this, ., number	1	2	0	0	6	3	18	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 18
this.ringing = true;	this, ., ringing, =, true	1	2	0	0	5	3	15	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 15
}									
}									
public void answer() {	void, answer()	1	2	0	0	2	3	6	No Encapsulation: P = 1. Inheritance: Q = 2

									No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
`if (ringing) {`	`if(), ringing`	1	2	0	0	2	3	6	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 6
`System.out.println ("Answering the call");`	`System, ., out, ., println(), "Answering the call"`	1	2	0	0	6	3	18	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 18
`ringing = false;`	`ringing, =, false`	1	2	0	0	3	3	9	No Encapsulation: P = 1. Inheritance: Q = 2 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 9
`}`									
`}`									
`@Override public void details() {`	`void, details()`	1	2	-1	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 4
`System.out.println ("This is a phone");`	`System, ., out, ., println(), "This is a phone"`	1	2	-1	0	6	2	12	No Encapsulation: P = 1. Inheritance: Q = 2 Polymorphism: R = -1 No Abstraction: S = 0 Z = 6 Sum = 12
`}`									
`}`									

public class LandlinePhone extends Phone {									
private int numberOfKeys;	int, numberOfKeys	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private boolean cliFeature;	boolean, cliFeature	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private String brand;	String, brand	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
private String model;	String, model	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
public LandlinePhone(int number) {	LandlinePhone(), int, number	1	1	0	0	3	2	6	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 6
super(number);	super(), number	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4

}									
public LandlinePhone(int number, int numberOfKeys, boolean cliFeature, String brand, String model) {	LandlinePhone(), int, number, int, numberOfKeys, boolean, cliFeature, String, brand, String, model	1	1	0	0	11	2	22	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 11 Sum = 22
super(number);	super(), number	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
this.numberOfKeys = numberOfKeys;	this, ., numberOfKeys, =, numberOfKeys	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
this.cliFeature = cliFeature;	This, ., cliFeature, =, cliFeature	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
this.brand = brand;	This, ., brand, =, brand	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10
this.model = model;	This, ., model, =, model	1	1	0	0	5	2	10	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 10

}									
public int getNumberOfKeys() {	int, getNumberOfKeys()	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
return numberOfKeys;	return, numberOfKeys	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
}									
public void setNumberofKeys(int numberOfKeys) {	Void, setNumberofKeys(), int, numberOfKeys	-1	1	0	0	4	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 0
this.numberOfKeys = numberOfKeys;	This, ., numberOfKeys, =, numberOfKeys	-1	1	0	0	5	-1	-5	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = -5
}									
public boolean isCliFeature() {	boolean, isCliFeature()	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
return cliFeature;	Return, cliFeature	1	1	0	0	2	2	4	No Encapsulation: P = 1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 4
}									

`public void setCliFeature(boolean cliFeature) {`	`void,`
	`setCliFeature(),`
	`Boolean,`
	`cliFeature`
		-1	1	0	0	4	0	0	Encapsulation: P = -1.
									Inheritance: Q = 1
									No Polymorphism: R = 0
									No Abstraction: S = 0
									Z = 4
									Sum = 0

`this.cliFeature = cliFeature;`	`This, .,`
	`cliFeature,`
	`=,`
	`cliFeature`
		-1	1	0	0	5	0	0	Encapsulation: P = -1.
									Inheritance: Q = 1
									No Polymorphism: R = 0
									No Abstraction: S = 0
									Z = 5
									Sum = 0

}									

`public String getBrand() {`	`String,`
	`getBrand()`
		-1	1	0	0	2	0	0	Encapsulation: P = -1.
									Inheritance: Q = 1
									No Polymorphism: R = 0
									No Abstraction: S = 0
									Z = 2
									Sum = 0

`return brand;`	`return,`
	`brand`
		-1	1	0	0	2	0	0	Encapsulation: P = -1.
									Inheritance: Q = 1
									No Polymorphism: R = 0
									No Abstraction: S = 0
									Z = 2
									Sum = 0

}									

`public void setBrand(String brand) {`	`void,`
	`setBrand(),`
	`String,`
	`brand`
		-1	1	0	0	4	0	0	Encapsulation: P = -1.
									Inheritance: Q = 1
									No Polymorphism: R = 0
									No Abstraction: S = 0
									Z = 4
									Sum = 0

`this.brand = brand;`	`This, .,`
	`brand, =,`
	`brand`
		-1	1	0	0	5	0	0	Encapsulation: P = -1.
									Inheritance: Q = 1
									No Polymorphism: R = 0
									No Abstraction: S = 0
									Z = 5
									Sum = 0

}									

`public String getModel() {`	`String,`
	`getModel()`
		-1	1	0	0	2	0	0	Encapsulation: P = -1.
									Inheritance: Q = 1
									No Polymorphism: R = 0
									No Abstraction: S = 0

									Z = 2 Sum = 0
return model;	return, model	-1	1	0	0	2	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 2 Sum = 0
}									
public void setModel(String model) {	void, setModel(), String, model	-1	1	0	0	4	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 0
this.model = model;	This, ., model, =, model	-1	1	0	0	5	0	0	Encapsulation: P = -1. Inheritance: Q = 1 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 5 Sum = 0
}									
@Override public void isRinging() {	void, isRinging()	1	1	-1	0	2	1	2	No Encapsulation: P = 1. Inheritance: Q = 1 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 2
if (ringing) {	if(), ringing	1	1	-1	0	2	1	2	No Encapsulation: P = 1. Inheritance: Q = 1 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 2
System.*out*.println ("Lanline phone is ringing");	System, ., *out*, ., println(), "Lanline phone is ringing"	1	1	-1	0	6	1	6	No Encapsulation: P = 1. Inheritance: Q = 1 Polymorphism: R = -1 No Abstraction: S = 0 Z = 6 Sum = 6
}									
}									
@Override	void, details()	1	1	-1	0	2	1	2	No Encapsulation: P = 1.

public void details() {									Inheritance: Q = 1 Polymorphism: R = -1 No Abstraction: S = 0 Z = 2 Sum = 2
System.out.println ("This is a " + brand + " " + model + " Landline Phone");	System, ., out, ., println(), "This is a ", +, brand, +, " ", +, model, +, " Landline Phone"	1	1	-1	0	14	1	14	No Encapsulation: P = 1. Inheritance: Q = Polymorphism: R = -1 No Abstraction: S = 0 Z = 14 Sum = 14
}									
}									
public interface Device {									
public void switchOn();	void, switchOn()	1	0	0	-1	2	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 Abstraction: S = -1 Z = 2 Sum = 0
public void switchOff();	void, switchOff()	1	0	0	-1	2	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 Abstraction: S = -1 Z = 2 Sum = 0
public void details();	void, details()	1	0	0	-1	2	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 Abstraction: S = -1 Z = 2 Sum = 0
}									
public class Main {									
public static void main(String[] args) {	Static, void, main(), String[], args	1	0	0	0	5	1	5	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0

									No Abstraction: S = 0 Z = 2 Sum = 5
Device device1 = new Phone("0112845678");	Device, device1, =, new, Phone(), "0112845678"	1	0	0	0	6	1	6	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 6
device1.details();	device1, . , details()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Polymorphism: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
Phone phone1 = new Phone("0778902245");	Phone, phone1, =, new, Phone(), "0778902245"	1	0	0	0	6	1	6	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 6
phone1.details();	phone1, . , details()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Polymorphism: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
Phone phone2 = new SmartPhone("0718900817","Nokia ", "Android 8.0", "5.5", "3000 mAh","Nokia", "6");	Phone, phone2, =, new, SmartPhone(), "0718900817", ,, "Nokia ", ,, "Android 8.0", ,, "5.5", ,, "3000 mAh", ,, "Nokia", ,, "6"	1	0	0	0	18	1	18	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 18 Sum = 18
phone2.details();	phone2, ., details()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Polymorphism: R = -1

									No Abstraction: S = 0 Z = 3 Sum = 0
phone2.call("0718900817");	phone2, ., call(), "0718900817"	1	0	0	0	4	1	4	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 4
phone2.isRinging();	phone2, ., isRinging()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Polymorphism: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
phone2.answer();	phone2, ., answer()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 3
SmartPhone smartPhone = new SmartPhone("0778902345","Reflection","ios","5.8","2700mAh","Apple","iPhone x");	SmartPhone, smartphone, =, new, SmartPhone(), "0778902345", ,, "Reflection", ,, "ios", ,, "5.8", ,, "2700mAh", ,, "Apple", ,, "iPhone x"	1	0	0	0	18	1	18	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 18 Sum = 18
smartPhone.details();	smartphone, ., details()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
smartPhone.call("0778902345");	smartphone, ., call(), "0778902345"	1	0	0	0	4	1	4	No Encapsulation: P = 1. Inheritance: Q = 0

									No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 4
smartPhone.isRinging();	smartphone, ., isRinging()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Polymorphism: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
smartPhone.answer();	smartphone, ., answer()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 3
smartPhone.sendMessage("0778902345", "hiii");	smartphone, ., sendMessage(), "0778902345", ,, "hiii"	1	0	0	0	6	1	6	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 6 Sum = 6
smartPhone.isMessaging();	smartphone, ., isMessaging()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 3
smartPhone.readMessage();	smartphone, ., readMessage()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 3
LandlinePhone landLinePhone = new LandlinePhone("01126456789");	LandlinePhone, landLinePhone, =, new, LandlinePhone(),	1	0	0	0	6	1	6	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0

	"01126456789"								No Abstraction: S = 0 Z = 6 Sum = 6
landLinePhone.setBrand("Tp-Link");	landLinePhone, ., setBrand(), "Tp-Link"	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = -4
landLinePhone.setModel("t6");	landLinePhone, ., setModel(), "t6"	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = -4
landLinePhone.setCliFeature(false);	landLinePhone, ., setCliFeature(), false	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = -4
landLinePhone.setNumberofKeys(20);	landLinePhone, ., setNumberofKeys(), 20	-1	0	0	0	4	-1	-4	Encapsulation: P = -1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = -4
landLinePhone.details();	landLinePhone, ., details()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Polymorphism: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
LandlinePhone landLinePhone2 = new LandlinePhone("01126456789",21,true,"Panasonic","R6");	LandlinePhone, landLinePhone2, =, new, LandlinePhone(), "01126456789" , ,, 21, ,, true, ,, "Panasonic", ,, "R6"	1	0	0	0	14	1	14	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 14 Sum = 14
landLinePhone.details();	landLinePhone, ., details()	1	0	-1	0	3	0	0	No Encapsulation: P = 1.

									Inheritance: Q = 0 Polymorphism: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
`landLinePhone2.call("01126456789");`	landLinePhone2, ., call(), "01126456789"	1	0	0	0	4	1	4	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 4 Sum = 4
`landLinePhone2.isRinging();`	landLinePhone2, ., isRinging()	1	0	-1	0	3	0	0	No Encapsulation: P = 1. Inheritance: Q = 0 Polymorphism: R = -1 No Abstraction: S = 0 Z = 3 Sum = 0
`landLinePhone2.answer();`	landLinePhone2, ., answer()	1	0	0	0	3	1	3	No Encapsulation: P = 1. Inheritance: Q = 0 No Polymorphism: R = 0 No Abstraction: S = 0 Z = 3 Sum = 3
}									
}									
WCC								**672**	

CONCLUSION

Complexity metrics can be used to predict the quality of a software system. A set of object-oriented factors are used to introduce the new object-oriented complexity metric. This report includes the explanations of how each factor considered in the new metric affect the complexity of a program and the explanations of how the new metric captures the complexity introduced by each factor. Each of the four factors were described considering important features.

Then the report includes the new object-oriented complexity metric which was introduced using those set of factors and the explanation of how the new metric was generated considering those factors. Four random Java programs are used to evaluate the complexity level using the newly introduced complexity metric. The newly proposed metric is calculated using source codes; therefore, this metric can be a good predictor of reusability, understandability, testing efforts and future maintenance efforts. This newly introduced metric can be used to easily develop a complexity measuring tool.

REFERENCES

[1] Raees Ahmad Khan, A. Yadav, “Development of Encapsulated Class Complexity Metric,” in Procedia Technology 4 (December 2012), 754 – 760

[2] D. Mishra and A. Mishra, Object-Oriented Inheritance Metrics in the Context of Cognitive Complexity (January 2011)

[3]Erez Metula, in Managed Code Rootkits, 2011 [Online]. Available: https://www.sciencedirect.com/topics/computer-science/polymorphism

[4]Hosk, August 12, 2015 [Online]. Available: https://crmbusiness.wordpress.com/2015/08/12/why-understanding-abstractions-helps-you-write-better-code/

[5] FreeCodeCamp, 5 September 2018 [Online]. Available: https://www.freecodecamp.org/news/make-your-code-understandable-by-using-abstraction-4b522307130c/

[6] BeginnersBook, [Online]. Available: https://beginnersbook.com/2014/01/abstract-method-with-examples-in-java/