**Day 10\_Java Assignment**

**1. Problem Description:**

What is polymorphism?

What are the types of polymorphism?

Give an example for each type of polymorphism.

**2. My Solution:**

The word Polymorphism means having many forms. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and having multiple implementations. Though the contract defined by the interface remains the same, each class implements the contract differently and thus exhibits different behavior. The word “poly” means many and “morphs” means forms, So it means many forms.

**Type of Polymorphism:**

* Compile time polymorphism (method overloading)
* Runtime polymorphism (method overriding)

**Compile time polymorphism:**

The flow of control is decided in compile time itself. It is achieved using method overloading.

In method overloading, an object can have two or more methods with the same name, but the method parameters are different such as:

* number of parameters are different
* method parameter types are different

**Example:**

**package** polymorphism;

**public** **class** Calculator {

**public** Integer sum(Integer a, Integer b) {

**return** a + b;

}

**public** Float sum(Float a, Float b) {

**return** a + b;

}

**public** Double sum(Double a, Double b) {

**return** a + b;

}

**public** **static** **void** main(String[] args) {

Calculator calc = **new** Calculator();

Integer sum1 = calc.sum(1, 2);

Float sum2 = calc.sum(1f, 2f);

Double sum3 = calc.sum(1d, 2d);

System.***out***.println(sum1);

System.***out***.println(sum2);

System.***out***.println(sum3);

}

}

Output:

3

3.0

3.0

**Runtime Polymorphism:**

Runtime Polymorphism is referred as method overriding when we extend a class into a child class. In runtime polymorphism, which method will be invoked is decided on the runtime based on the actual instance of the child class.

**Example:**

**package** polymorphism;

**class** Animal1 {

**public** **void** makeNoise() {

System.***out***.println("Some sound");

}

}

**class** Dog **extends** Animal1 {

@Override

**public** **void** makeNoise() {

System.***out***.println("Bark");

}

}

**class** Cat **extends** Animal1 {

@Override

**public** **void** makeNoise() {

System.***out***.println("Meawoo");

}

}

**public** **class** Animal2 {

**public** **static** **void** main(String[] args) {

Animal1 cat = **new** Cat();

cat.makeNoise();

Animal1 dog = **new** Dog();

dog.makeNoise();

}

}

**Output:**

Meawoo

Bark