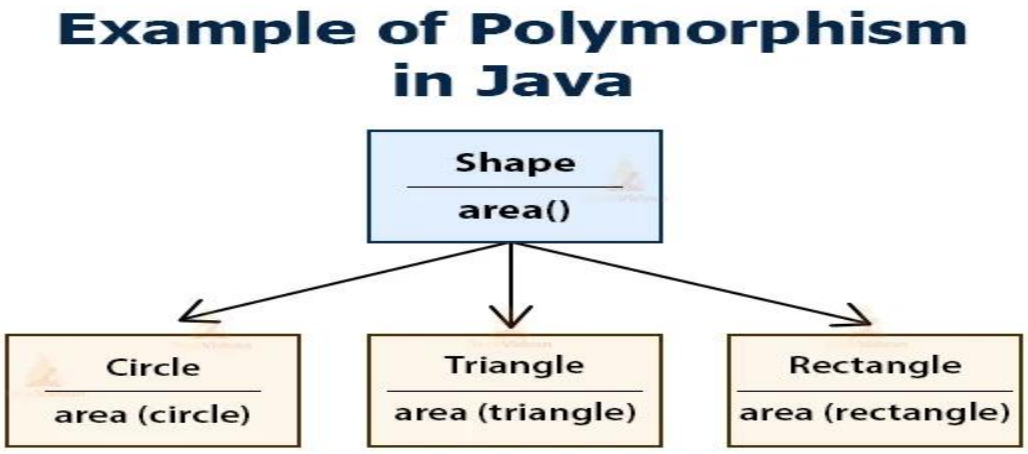
Khulna Khan Bahadur Ahsanullah University

Object-oriented programming

CSE 1203

1. What is Polymorphism?

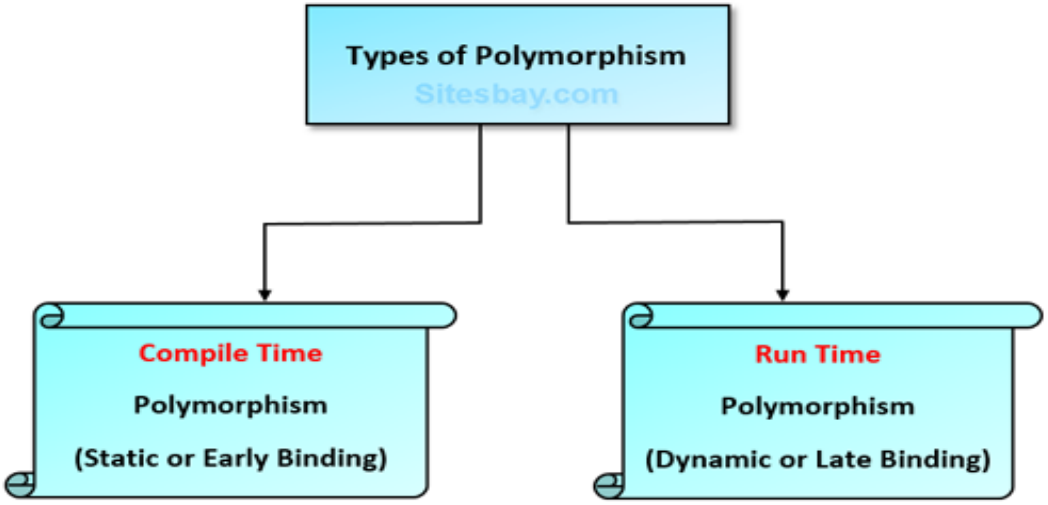
- When one task is performed by different ways, it is known as polymorphism.
- We use method overloading and method overriding to achieve polymorphism.
- Example: If same message is passed to many students and students answer different way.



Advantage of method overloading

- Method overloading *increases the readability of the program*.
- Overloaded methods give programmers the flexibility to call a similar method for different types of data.
- Overloading is also used on constructors to create new objects given different amounts of data

Types of Polymorphism



- Compile time Polymorphism-It is also known as static polymorphism. This type of polymorphism is **achieved by method overloading**.
- Run time Polymorphism-It is also known as Dynamic polymorphism. This type of polymorphism is **achieved by Method Overriding.**

2. Method overloading

- If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.
- If we have to perform only one operation, having same name of the methods increases the readability of the program.
- Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs.

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

> In Java, Method Overloading is not possible by changing the return type of the method only.

1) **Method Overloading: changing no. of arguments**

- In this example, we have created two methods, first add() method performs addition of two numbers and second add method performs addition of three numbers.
- In this example, we are creating static methods so that we don't need to create instance for calling methods.

```
1. class Adder{
2. static int add(int a,int b){return a+b;}
3. static int add(int a,int b,int c){return a+b+c;}
4. }
5. class TestOverloading1{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(11,11,11));
9. }}
```

Output:

```
22
33
```

2) Method Overloading: changing data type of arguments

- In this example, we have created two methods that differs in data type.
- The first add method receives two integer arguments and second add method receives two double arguments.

```
1. class Adder{
2. static int add(int a, int b){return a+b;}
3. static double add(double a, double b){return a+b;}
4. }
5. class TestOverloading2{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(12.3,12.6));
9. }}
```

Output:

```
22
24.9
```

Q) Why Method Overloading is not possible by changing the return type of method only?

In java, method overloading is not possible by changing the return type of the method only because of ambiguity. Let's see how ambiguity may occur:

```
1. class Adder{
2. static int add(int a,int b){return a+b;}
3. static double add(int a,int b){return a+b;}  }
```

```
4. class TestOverloading3{
5. public static void main(String[] args){
6. System.out.println(Adder.add(11,11));//ambiguity
7. }}
```

Output:

```
Compile Time Error: method add(int,int) is already defined in class Adder
```

> Note: Compile Time Error is better than Run Time Error. So, java compiler renders compiler time error if you declare the same method having same parameters.

Can we overload java main () method?

Yes, by method overloading. You can have any number of main methods in a class by method overloading. But JVM calls main () method which receives string array as arguments only. Let's see the simple example:

```
1. class TestOverloading4{
2. public static void main(String[] args){System.out.println("main with String[]")
   ;}
3. public static void main (String args){System.out.println("main with String");}
4. public static void main(){System.out.println("main without args");}
5. }
```

Output:

```
main with String []
```

Method Overriding

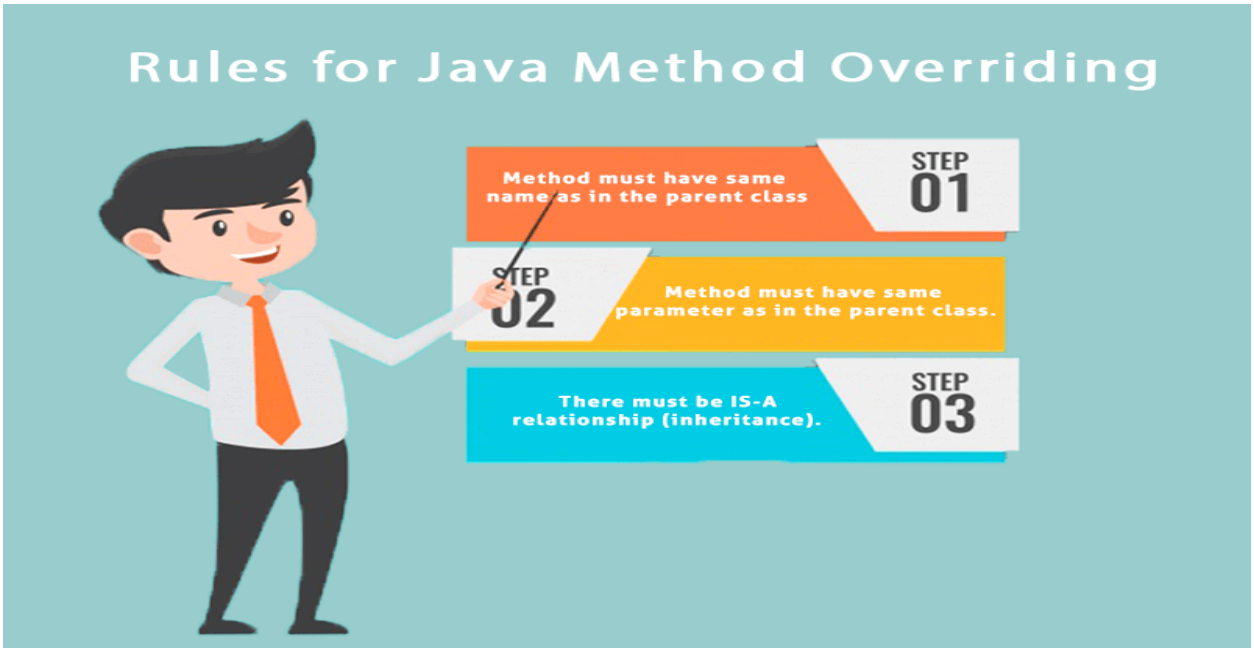
- If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.
- In other words, if a subclass provides the specific implementation of the method that has been declared by one of its parent classes, it is known as method overriding.

Usage of Java Method Overriding

- Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.
- Method overriding is used for runtime polymorphism

Rules for Java Method Overriding

1. The method must have the same name as in the parent class
2. The method must have the same parameter as in the parent class.
3. There must be an IS-A relationship (inheritance).

- 

Example of method overriding

- In this example, we have defined the run method in the subclass as defined in the parent class but it has some specific implementation.
- The name and parameter of the method are the same, and there is IS-A relationship between the classes, so there is method overriding.

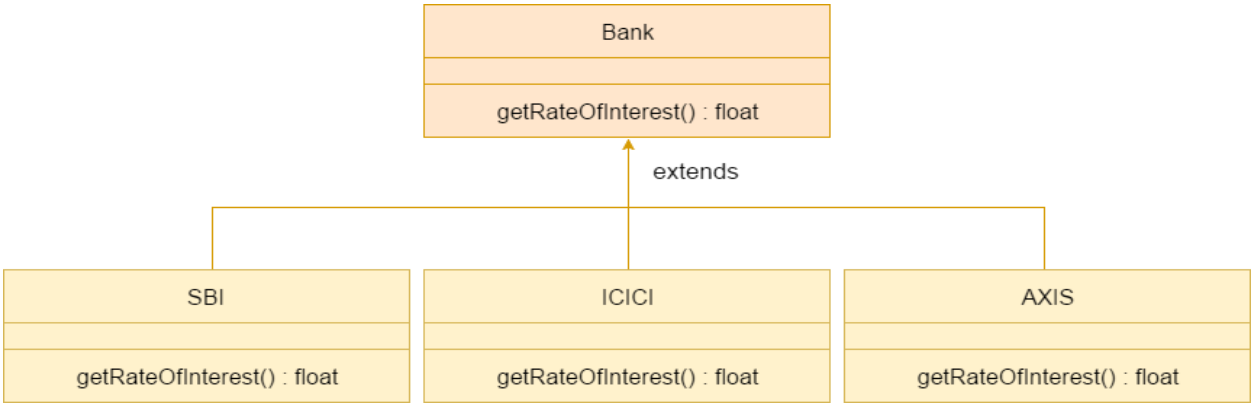
```
1. //Java Program to illustrate the use of Java Method Overriding
2. //Creating a parent class.
3. class Vehicle {
4.   //defining a method
5.   void run () {System.out.println("Vehicle is running");}
6. }
7. //Creating a child class
8. class Bike2 extends Vehicle {
9.   //defining the same method as in the parent class
10. void run () {System.out.println("Bike is running safely");}
11.
12. public static void main(String args[]){
13. Bike2 obj = new Bike2();//creating object
14. obj.run();//calling method
15. }
16.}
```

Output:

```
Bike is running safely
```

A real example of Java Method Overriding

- Consider a scenario where Bank is a class that provides functionality to get the rate of interest. However, the rate of interest varies according to banks.
- For example, SBI, ICICI and AXIS banks could provide 8%, 7%, and 9% rate of interest.




```
1. //where three classes are overriding the method of a parent class.
2. //Creating a parent class.
3. class Bank{
4. int getRateOfInterest(){return 0;}
5. }
6. //Creating child classes.
7. class SBI extends Bank{
8. int getRateOfInterest(){return 8;}
9. }
10.class ICICI extends Bank{
11.int getRateOfInterest(){return 7;}
12.}
13.class AXIS extends Bank{
14.int getRateOfInterest(){return 9;}  }
```

```
15.//Test class to create objects and call the methods
16.class Test2{
17.public static void main(String args[]){
18.SBI s=new SBI();
19.ICICI i=new ICICI();
20.AXIS a=new AXIS();
21.System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());
22.System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());
23.System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());
24.}  }
```

```
Output:
SBI Rate of Interest: 8
ICICI Rate of Interest: 7
AXIS Rate of Interest: 9
```

Can we override static method?

No, a static method cannot be overridden. It can be proved by runtime polymorphism.

Why can we not override static method?

It is because the static method is bound with class whereas instance method is bound with an object. Static belongs to the class area, and an instance belongs to the heap area.

Can we override java main method?

No, because the main is a static method.

Difference between method overloading and method overriding

A list of differences between method overloading and method overriding are given below:

No.	Method Overloading	Method Overriding
1)	Method overloading is used *to increase the readability* of the program.	Method overriding is used to provide the specific implementation of the method that is already provided by its super class.
2)	Method overloading is performed *within class*.	Method overriding occurs *in two classes* that have IS-A (inheritance) relationship.
3)	In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
4)	Method overloading is the example of *compile time polymorphism*.	Method overriding is the example of *run time polymorphism*.
5)	In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter.	*Return type must be same or covariant* in method overriding.

Java Method Overloading example

```
1. class OverloadingExample{
2. static int add(int a,int b){return a+b;}
3. static int add(int a,int b,int c){return a+b+c;}
4. }
```

Java Method Overriding example

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void eat(){System.out.println("eating bread...");}
6. }
```