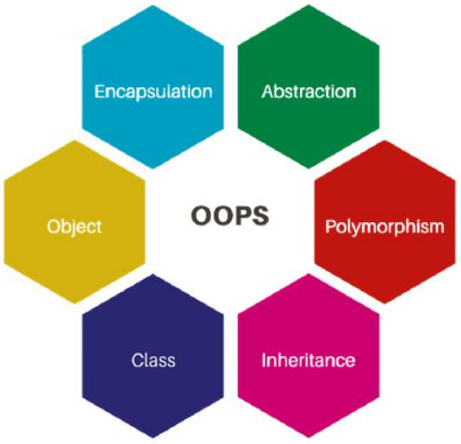


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Java™

Polymorphism

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Polymorphism

- The word polymorphism can be broken down into Poly and morphs, as "**Poly**" means **many** and "**Morphs**" means **forms.**
- In simple words, we can say that ability of a message to be **represented in many forms**.
- **carbon** can exist in many forms, i.e., **diamond, graphite, coal**, etc.
- a **lady** can have different characteristics simultaneously. She can be a **mother, a daughter, or a wife.**

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Polymorphism

- **Polymorphism in Java is mainly divided into two types.**
  - Compile-time polymorphism
  - Runtime polymorphism
- Compile-time polymorphism can be achieved by **method overloading**
- Runtime polymorphism can be achieved by **method overriding**.

3

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Polymorphism

- **Compile-time polymorphism**
  - This type of polymorphism in Java is also called **static polymorphism** .
  - It can be achieved by method overloading. In this process, an overloaded method is resolved at compile time rather than resolving at runtime.

4

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Polymorphism -Method overloading

- Consider a class where multiple methods have the same name. It will be difficult for the compiler to distinguish between every method.
- To overcome this problem, we pass a different number of arguments to the method or different types of arguments to the method. In this way, we achieve method overloading.
- In other words, a class can have multiple methods of **the same name, and each method can be differentiated either by bypassing different types of parameters or bypassing a different number of parameters.**

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Method Overloading

- If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.
- Why Method Overloading?
  - It is used when different objects are required to perform a similar set of tasks but use different input parameters.
- Best Example - println() method.
  - System.out.println(),
  - System.out.println(int)
  - System.out.println(double)
  - System.out.println(string)
  - System.out.println(character)

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Two ways to overload the method

- By changing the number of arguments
- By changing the data type

```
class Adder{
        int add(int a,int b){
                return a+b;
        }
        int add(int a,int b,int c){
                return a+b+c;
        }
}
```

```
class Adder{
        int add(int a,int b){
                return a+b;
        }
        double add(double a, double b) {
                return a+b;
        }
}
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Can we overload java main() method?

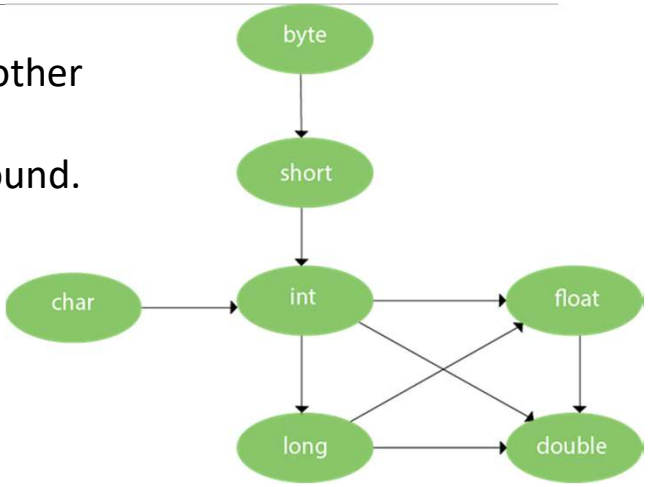
- Yes, by method overloading.
- But JVM calls main() method which receives string array as arguments only.

- **public static void** main(String[] args)
- **public static void** main(String args)
- **public static void** main(int args)

Method Overloading and Type Promotion

ABESIT COLLEGE OF ENGINEERING
CRC-Training

One type is promoted to another implicitly,
if no matching datatype is found.



ABESIT COLLEGE OF ENGINEERING
CRC-Training

Example - Overloading with TypePromotion

```
class cal{
    int add(int a, int b,int c){return a+b+c;}
    double add(int a, double b){return a+b;}
}
public class Main
{
    public static void main(String[] args) {
        cal obj1 = new cal();
        cal obj2 = new cal();
        System.out.println(obj1.add(10,20));
        System.out.println(obj1.add(10,20.5));
    }
}
```

now second int literal will be promoted to double

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Type Promotion in case of ambiguity

```
class cal{
    double add(double a, int b){return a+b;}
    double add(int a, double b){return a+b;}
}
public class Main
{
    public static void main(String[] args) {
        cal obj1 = new cal();
        System.out.println(obj1.add(10,20.5));
        System.out.println(obj1.add(10.5,20));
    }
}
```

```
30.5
30.5
```

What will happen when we call add a method like –

obj1.add(10,10);

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Type Promotion in case of ambiguity

```
1  class cal{
2      double add(double a, int b){return a+b;}
3      double add(int a, double b){return a+b;}
4  }
5  public class Main
6  {
7      public static void main(String[] args) {
8          cal obj1 = new cal();
9          System.out.println(obj1.add(10,10));
10     }
11 }
```

Compilation failed due to following error(s).

```
Main.java:9: error: reference to add is ambiguous
                System.out.println(obj1.add(10,10));
                                   ^
  both method add(double,int) in cal and method add(int,double) in cal match
1 error
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Runtime Polymorphism

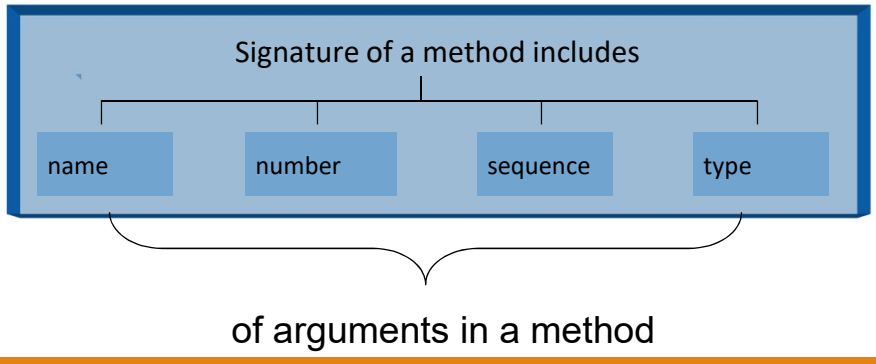
- Runtime polymorphism is also called Dynamic method dispatch. Instead of resolving the overridden method at compile-time, it is **resolved at runtime.**
- Here, an overridden child class method is called through its parent's reference.
- Then the method is evoked according to the type of object.
- In runtime, JVM figures out the object type and the method belonging to that object.

13

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Method Overriding

- Method overriding is defined as creating a method in the subclass that has the same return type and signature as a method defined in the superclass.



14

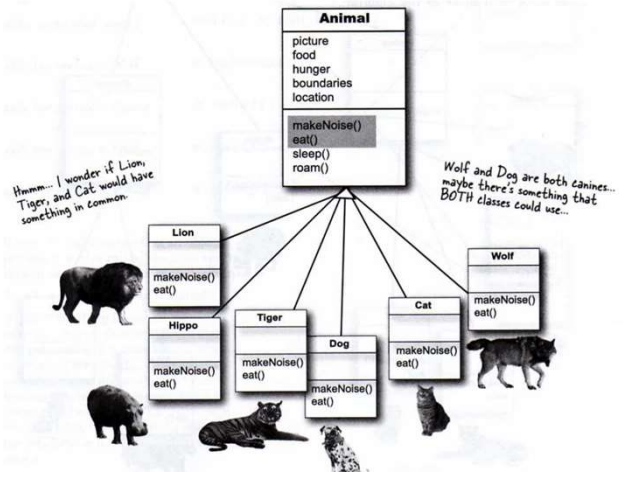
ABESIT COLLEGE OF ENGINEERING
CRC-Training

Method Overriding

- You can override (most) methods in a parent class by defining a method with the same name and parameter list in the child class.
- This is useful for changing the behavior of a class without changing its interfacse.
- You cannot override the static and final methods of a superclass.
- A subclass must override the abstract methods of a superclass.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Method Overriding



ABESIT COLLEGE OF ENGINEERING
CRC-Training

Method Overriding

- Any time you have a class that inherits a method from a superclass, you have the opportunity to override the method.

Example:

```
class Animal            {
    public void eat()
    {
   System.out.println("Generic Animal eating Generically ");
            }
     }
class Dog extends Animal    {
            public void eat()
            {
              System.out.println("EAT MEAT");
            }
      }
class Cow extends Animal
{
    public void eat()
    {
        System.out.println("EAT GRASS ");
    }
}
```

Here eat() method overridden

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Overriding Vs Overloading

Any time you have a class that inherits a method from a superclass, you have the opportunity to override the method.	Overloaded methods let you reuse the same method name in a class, but with different arguments(and optionally , a different return type.
The argument list must be same.	Overloaded methods must change the argument list.
The return type must be same in jdk1.4, but in jdk1.5 it should be co-variant returns	Overloaded methods can change the return type.
The access level can't be less restrictive in Overridden method.	Overloaded methods can change the access modifier.
Can throw any unchecked exception by Overriding. The Overridden method can throw any sub type or same type exception of overriding method.	Overloaded methods can declare new or broader checked exceptions.
Without inheritance no Overriding.	A method can be overloaded in the same class or in a subclass.
You cannot override a method marked final. You cannot override a method marked static.	Can overload final and static methods.

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

1) What will be the output of the following code?

```
class A
{
      void show()
      {
            System.out.println(" A class Show Call ");
      }
}
class B extends A
{
      String show()
      {
            System.out.println("B Class show call ");
      }
}
            B obj=new B();      obj.show();
```


ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

2) What is this , Overloading or Overriding?

```
class A
{
      void show()
      {
            System.out.println(" A class Show Call ");
      }
}
class B extends A
{
      void show(int i)
      {
            System.out.println("B Class show call ");
      }
}
```

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

3) What will be the output of the following code?

```
class A {
      public void show()
      {
            System.out.println(" A class Show Call ");
      }
}
class B extends A       {
      void show()
      {
            System.out.println("B Class show call ");
      }
}
```

B obj=new B(); obj.show();

ABESIT COLLEGE OF ENGINEERING
CRC-Training

Test Your Self

4) What will be the output of the following code?

```
class A   {
      private final void show()
      {
            System.out.println(" A class Show Call ");
      }
}
class B extends A       {
      void show()
      {
            System.out.println("B Class show call ");
      }
}
```

B obj=new B(); obj.show();