

Topics



- Method Signatures
- Polymorphism
- Polymorphism Types



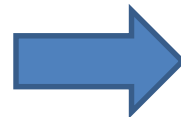
Method Signatures

- A Method Signature is the method name and the number, type and order of its parameters .
- Return type of a method is not considered to be a part of the method signature.

Methods

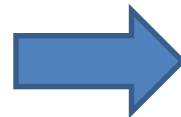
```
int    doSomething(int a , float b)
{
    .....
}
```

```
float  doSome(int a , int b, int c)
{
    .....
}
```



Method Signatures

doSomething(int,float)



doSome(int,int,int)

Polymorphism



[https://en.wikipedia.org/wiki/Polymorphism_\(computer_science\)](https://en.wikipedia.org/wiki/Polymorphism_(computer_science))

- One Interface Having Multiple Forms
- Three Flavors
 1. Adhoc Polymorphism (Via Method Overloading)
 2. Subtyping (Via Method Overriding)
 3. Parametric Polymorphism (Via Generic Programming –Will be Covered Later)

Adhoc Polymorphism : Method Overloading



- Supported via Method Overloading
- Two Methods are said to be overloaded if they have same name but different signatures
- Method signatures can be different either via having different number of arguments to methods or via having different order of the arguments
- Overloaded method either may have same or different return types

int add(int a , int b) {} ➡ add(int , int)

float add(float a, float b) {} ➡ add(float , float)

double add(double a, double b) {} ➡ add(double , double)

String add(String a, String b) {} ➡ add(String , String)

Adhoc Polymorphism : Method Overloading



- Method Overloading Example

```
class MethodOverloadingExample
{
    int    doS(int a, float b)          { ... } → doS(int , float)
    float  doS(float a, int b)          { ... } → doS(float, int)
    int    doS(int a, float b, int c)   { ... } → doS(int , float, int)
    float  doS(int a, float b, double c){ ... } → doS(int , float, double)
}

float    doS(double a, float b)  { ... }
double   doS(double x, float y)  { ... }
```

Wrong Method Declaration in
In Same Class → Compile
Time Error
(Not Overloaded Methods)

Polymorphism: Subtyping (Via Method Overriding)



- This Type of Polymorphism is known as Runtime Polymorphism (Dynamic Method Dispatch or Method Overriding)
- Two Methods are said to be overridden if and only if they have **same name**, **same signatures** and **same return type**
- Exhibited only by sub-classes of a common super class
- A sub class can override a method of a super class

```
class A
{
    public void doS(int a, int b) { .... }
} // End of class A
```

```
class B extends A
{
    public void doS(int a, int b) { .... }
} // End of class B
```

class B overrides the
doS() method of super
class A.

Polymorphism: Subtyping (Via Method Overriding)



- Method Overriding Example-1

```
class A
{
    public void print()
    {
        System.out.println("Hello Class A");
    }
}
// End of class A
```

```
class B extends A
{
    public void print()
    {
        System.out.println("Hello Class B");
    }
}
// End of class B
```

class B overrides the print() method of super class A.

```
A a1 = new A();
a1.print();
```

// What's the o/p?

```
a1 = new B();
a1.print();
```

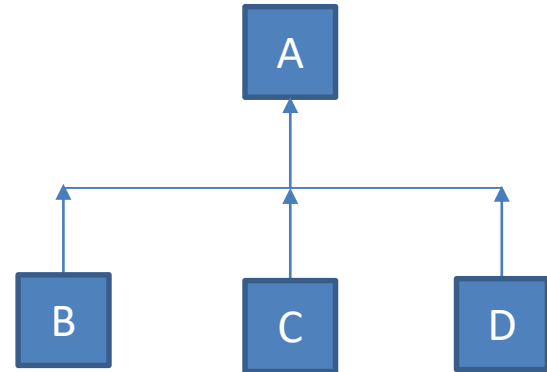
// What's the o/p?

Polymorphism: Subtyping (Via Method Overriding)



- Method Overriding Example-2

```
class A
{
    public void print()
    {
        System.out.println("Hello Class A");
    }
}
// End of class A
class B extends A
{
    public void print()
    {
        System.out.println("Hello Class B");
    }
}
// End of class B
class C extends A
{
    public void print()
    {
        System.out.println("Hello Class C");
    }
}
// End of class C
class D extends A
{
    public void print()
    {
        System.out.println("Hello Class D");
    }
}
// End of class D
```



Sub-classes B, C and D overrides the print() from the super class A

```
A a1 = new A();
a1.print();
```

```
a1 = new D();
a1.print();
```


Polymorphism: Subtyping (Via Method Overriding)



- Overridden Methods Cannot Have Different Return Types

class A

{

`public int print()`

{

`System.out.println("Hello Class A");`
`return 0;`

}

}// End of class A

class B extends A

{

`public void print()`

{

`System.out.println("Hello Class B");`

}

}// End of class B

Wrong ..
Compile Time
Error

Overridden
Methods Can
Not have
Different
Return Types

Polymorphism: Subtyping (Via Method Overriding)



- What is the problem with the following code?

```
class A
{
    public int print(int x)
    {
        System.out.println("Hello Class A");
        return 0;
    }
}
// End of class A
class B extends A
{
    public void print()
    {
        System.out.println("Hello Class B");
    }
}
// End of class B
```

Method Overloading

No Error → Not Method Overriding

Thank You