

Constructors

- ① Default Constructor → when no constructor is declared
 - ② Parameterized Constructor
 - ③ Copy Constructor
- ↳ default is not always executed

Polymorphism > Method Overloading

Many Forms

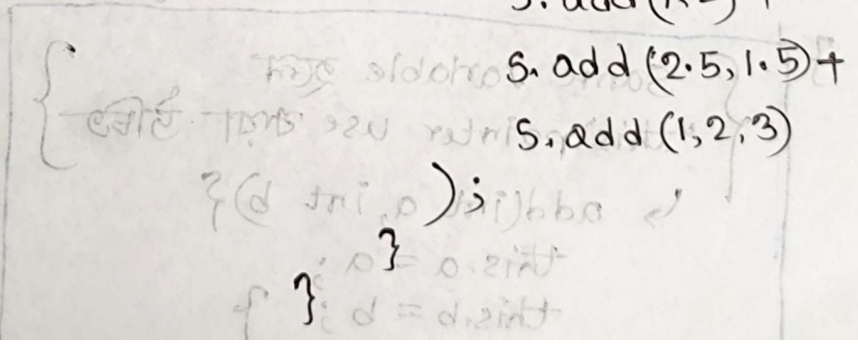
Overloading Identifiers

- ↳ Number of parameters and/or
- ↳ parameter types
- <type signature>

- ↳ return type different
- ↳ uniquely identify
- ↳ possible

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

```
public class Main {
    public static void main(String[] args) {
        sum s = new sum();
        System.out.println(
            s.add(1,2) +
            s.add(2.5, 1.5) +
            s.add(1,2,3)
        );
    }
}
```



#Take Object as Parameter

```
class add {  
    int a, b;  
    add(int a, int b) {  
        this.a = a;  
        this.b = b;  
    }  
}
```

access modifier
default
{ this pointer }

```
boolean get(add ob) {
```

```
    if (ob.a == a && ob.b == b) return true;
```

```
    return false;
```

```
}
```

```
public class Main {
```

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

```
        add ob1 = new add(5, 6);
```

```
        add ob2 = new add(10, 100);
```

```
        add ob3 = new add();
```

will cause error

```
        System.out.println(ob1.get(ob2));
```

↓
दुई object same memory

point करि ।

```
}  
    { Same variable छैन  
      "this" pointer use गरि छैन }  
    {  
        add(int a, int b) {  
            this.a = a;  
            this.b = b;  
        }  
    }  
}
```


#Returning Object

add ^{back}~~get~~() { ... }

add ^{back}~~get~~(add ob) {

add temp = new add(3,4);

temp.a = a + ob.a;

temp.b = b + ob.b;

return temp;

}

public class Main {

public static void main(String[] args) {

add ob3 = ob2.back(ob2);

System.out.

println(ob3.a + " "

+ ob3.b);

① declare a method
that take an object as
parameter

② declare a method
that returns an object

```
class Rectangle {
```

```
    double area;
```

```
    Rectangle(double area) {
```

```
        this.area = area;
```

```
    }
```

```
    Rectangle get(Rectangle r1, Rectangle r2) {
```

```
        Rectangle temp = new Rectangle(1);
```

```
        temp.area = r1.area * r2.area;
```

```
        return temp;
```

```
    }
```

```
public class Main {
```

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

```
        Rectangle rX = new Rectangle(1);
```

```
        Rectangle r1 = new Rectangle(15);
```

```
        Rectangle r2 = new Rectangle(2);
```

```
        System.out.println(rX(r1, r2));
```

```
        Rectangle r3 = rX.get(r1, r2);
```

```
        System.out.println(r3.area);
```


Array of Objects

```
class Student {  
    int id;  
    String name;  
  
    Student(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
  
    void show() {  
        System.out.println(  
            "id: " + id +  
            "name: " + name  
        );  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Student s[] = new Student [3];  
        ↑ type   ↑ array name   ↑ new Keyword   ↑ type   ↑ no. of objects
```

<1st class ka 3rd instance or object>

```
for(int i=0; i<s.length; i++) {
```

```
    s[i].show(123, "Rahim");
```

```
}    ↪ s[i] = new Student (123, "Rahim");
```

```
for(student s1: s) {
```

```
    s1.show();
```

```
}
```

```
Student() { id=45;
```

```
    name="Karim"; }
```

```
Student s[] = {
```

```
    new Student(),
```

```
    new Student(123, "Rahim");
```

```
    new Student(13, "ABC");
```

```
};
```

```
for (student s1: s) {
```

```
    s1.show();
```

```
}
```