Java

Agenda

- Class & Object
- Java Methods
- Methods Vs Constructor
- Java Constructor
- Method overloading

Java Class & Object

Class

```
class Employee
{
        int eid;
        String ename;
        double sal;
        String job;

        void display()
        {
        System.out.println(eid);
        System.out.println(ename);
        System.out.println(sal);
        System.out.println(job);
        }
        void bonus()
        {
        System.out.println((sal *10) /100);
        }
}
```

Variables

Methods

Object1

```
Employee emp1=new Employee();
emp1.eid=1020;
emp1.ename="John";
emp1.sal=80000;
emp1.job="Manager";

emp1.display();
```

Object2

```
Employee emp2=new
Employee();
emp2.eid=1021;
emp2.ename="David";
emp2.sal=50000;
emp2.job="Tech Assistant";

emp2.display();
```

Class & Object

- main() within class

```
class Student{
 int id=101;  //field or data member or instance variable
 String name="Anil";

 public static void main(String args[]){
  Student s1=new Student();//creating an object of Student
  System.out.println(s1.id);//accessing member through reference variable
  System.out.println(s1.name);
 }
}
```

Class & Object

- main() outside class

- In real time development, we create classes and use it from another class. It is a better approach than previous one.

- We can have multiple classes in different java files or single java file.

Student.java

```
class Student
{
 int id=101;
 String name="Anil";
}
```

Student1.java

```
Class Student1{
 public static void main(String args[])
  {
  Student s1=new Student();
  System.out.println(s1.id);
  System.out.println(s1.name);
  }
}
```

Class & Object

- 3 ways to initialize object variables in java.
  - By reference variable
  - By method
  - By constructor

Class & Object

- Initialization through reference variable

Student.java

```
class Student{
 int id;
 String name;
}
```

Student2.java

```
Class Student2{
 public static void main(String args[]){
  Student s=new Student();
  s.id=101;
  s.name="Anil";
  System.out.println(s.id+" "+s.name);
   }
}
```

Class & Object

- Initialization through method

Student.java

```
class Student {
 int id;
 String name;
 void insertRecord(int i, String n)
 {
  id=i;
  name=n;
 }
 void displayInformation()
  {
  System.out.println(id+" "+name);
  }
}
```

Student3.java

```
Class Student3{
 public static void main(String args[])
 {
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

Class & Object

- Initialization through constructor

Student.java

```
class Student{
 int id;
 String name;
 void  Student(int i, String n)
 {
  id=r;
  name=n;
 }
 void displayInformation()
 {
 System.out.println(id+" "+name);
 }
}
```

Student4.java

```
class TestStudent4
{
 public static void main(String args[])
 {
  Student s1=new Student(111,"Kiran");
  Student s2=new Student(222,"arya");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

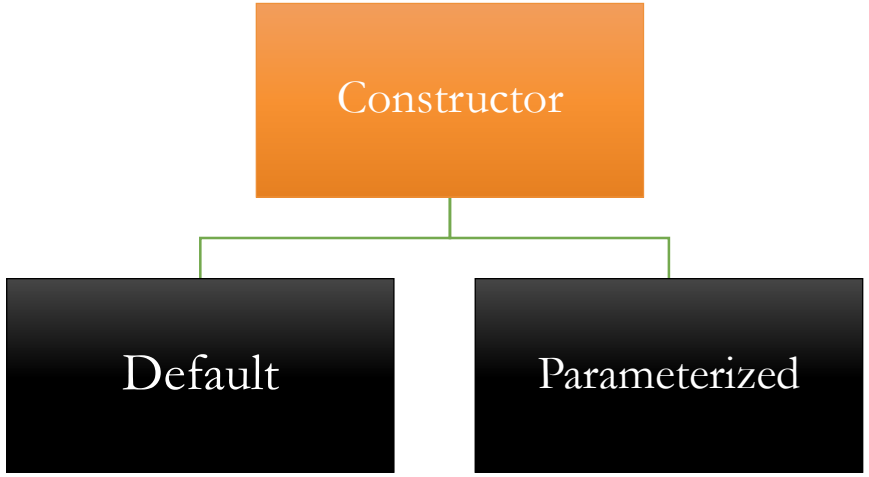
Java Methods

- A **method** is a set of code which is referred to by name and can be called (invoked) at any point in a program simply by utilizing the **method's** name.
- A **method** as a subprogram that acts on data and often returns a value.
- Each **method** has its own name.

	Parameter/s	Returned Value
Case1	✗	✗
Case2	✗	✓
Case3	✓	✗
Case4	✓	✓

Java Constructor

- **Constructor** in java is a special type of method that is used to initialize the object.
- Java constructor is invoked at the time of object creation.
- **Rules for creating java constructor:**
  1. Constructor name must be same as its class name.
  2. Constructor must have no explicit return type.
- There are 2 Types of Constructors.



Method V/s Constructor

Method

- Method name can be anything.
- Method can return a value.
- Need to call method explicitly.

Constructor

- Constructor name must be same as class name.
- Constructor doesn't return a value.
- Automatically invoked at the time of object creation.

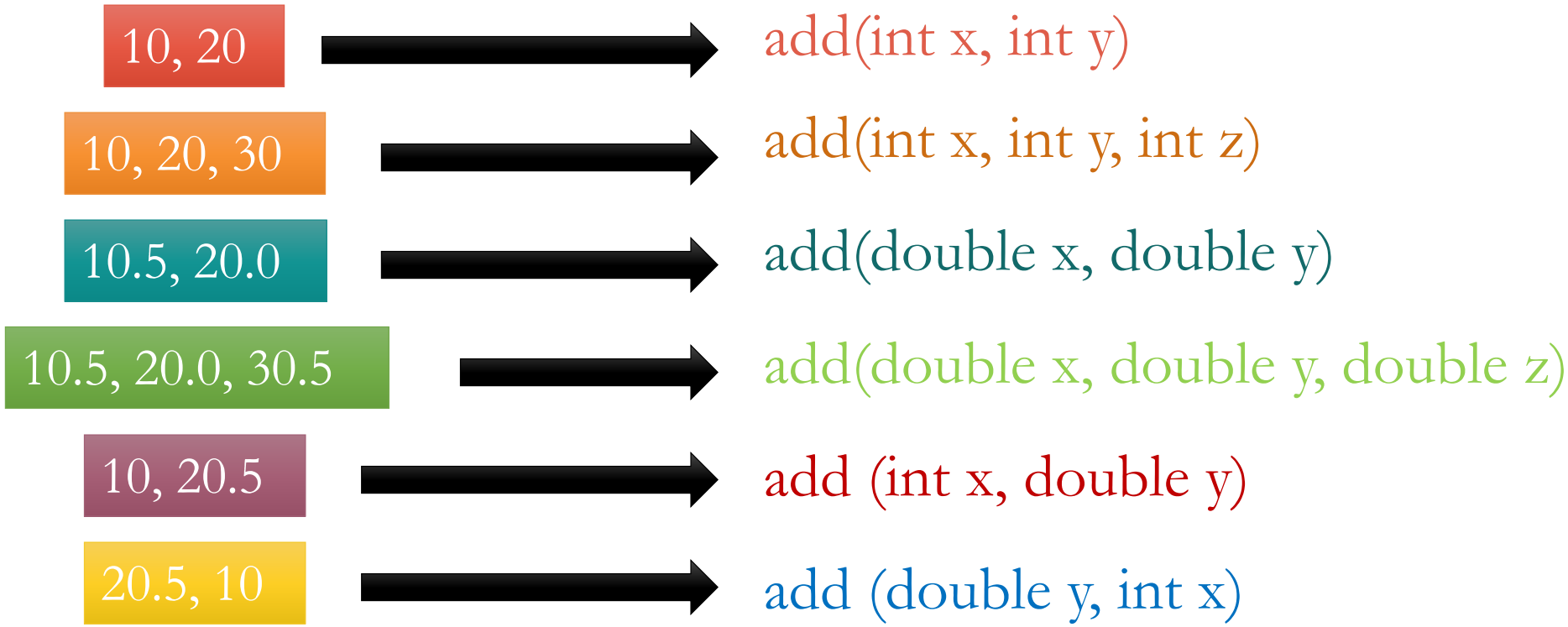
Method Overloading

- Method Overloading in Java is a concept related to Object Oriented Programming (OOP). Java supports overloading of methods and can distinguish between different methods with method signatures. A situation, wherein, in the same class there are two or more methods with same name, having different functions or different parameters, it is called Method Overloading.

- **Why Method Overloading?**
  - Using Method Overloading in Java is very common among Java programmers, because it:
  - Provides flexibility to call similar method for different data types
  - Saves memory
  - Saves time

- **Method Overloading can be done in two ways:**
  - By changing Arguments' data types
  - By changing number of Arguments

Overloading



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.

```
class TestOverloading
{
public static void main(String[] args)
{
System.out.println("main with String[]");
}
public static void main(String args)
{
System.out.println("main with String");
}
public static void main()
{
System.out.println("main without args");
}
}
```

Assignment

1. **Create a Student class contains the following variables and methods.**
   - **Class Name**: Student
   - **Variables** : SID , Sname, Sub1,Sub2,Sub3
   - **Methods:**
     - getStuData() Takes student details SID and Sname as parameters and assign them to variables.
     - getStuMarks() Takes student marks as parameters and assign them to Sub1, Sub2, Sub3.
     - totalMarks() Calculate total marks and print the student details with total marks.
   - Now, create objects from Student class stu1, stu2 etc. Then call Student class methods.
2. **Write a program to demonstrate constructor.**
   - Create a class 'Calculation' with 3 integer variable.
   - Create a constructor for assign the values into variables.
   - Then create another method 'sum' to calculate sum of 3 numbers.
   - Now, create object and call constructor by passing 3 integer values then call sum method.