OOP Concept

Polymorphism

Polymorphism means if same thing perform different behavior in different situation called as polymorphism.

Example1: Mobile is an best example of polymorphism because you can use mobile as phone as TV as location guide as radio device etc Means your single device work a different device means single device can change its behavior according to use requirement called as polymorphism.

Example2: If we think about person he change its behavior according to his requirement if person married then with wife behave like as husband, with parents behave like as child and with own child behave like as parent,in office behave like as employee with friend behave like as friend means we can say single person can change its behavior according to its requirement called as polymorphism.

If we want to work with polymorphism programmatically we have two ways
1. Compile time polymorphism:
  Q. What is compile time polymorphism?

Compile time polymorphism means object bounded with functionality at program compile time called as compile time polymorphism
2. Runtime polymorphism: Runtime polymorphism means object bounded or functionality bounded with object at program run time called as run time polymorphism
Note: Runtime polymorphism concept we will discuss in inheritance chapter at the time of method overriding concept.
Now we want to discuss about the compile time polymorphism

Q. How to implement compile time polymorphism practically using programming languages?

You can implement the compile time polymorphism using a function overloading.

Q. What is function overloading?

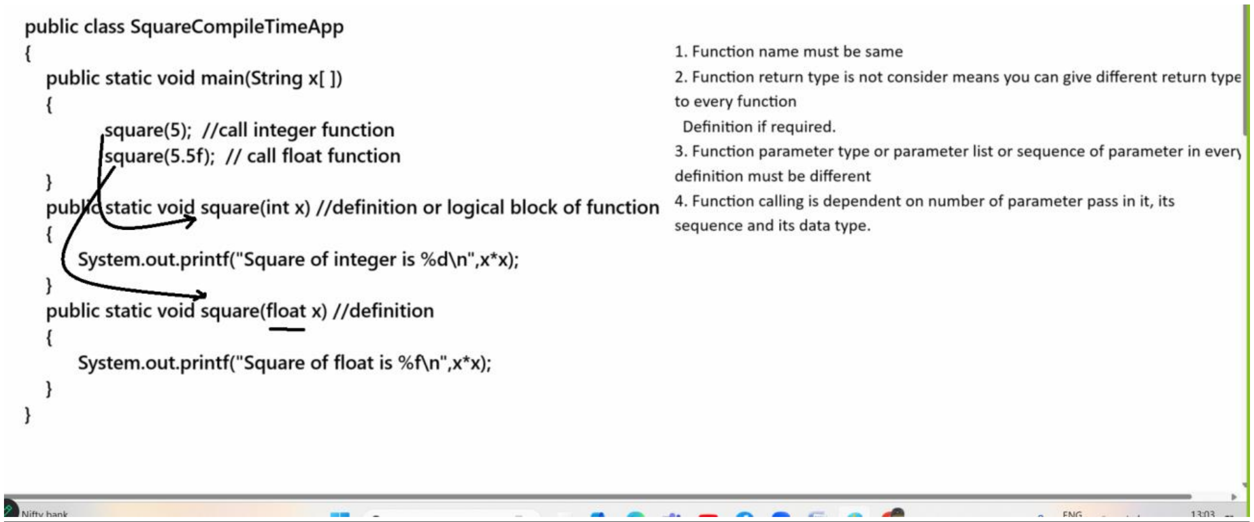
Function overloading means if we define multiple functions using same name with different parameter, with different parameter list with a different parameter sequence called as function overloading.
Example:
```
void add(int x,int y) :
void add(float x,float y)
void add(double x,double y)
void add(int x,float y)
void add(float x,int y)
```

etc

Note: if we think about above code we can say we have all function names as add and every function has different data type or different parameter list or different parameter sequence so we can say above all declaration say it is implementation of function overloading.

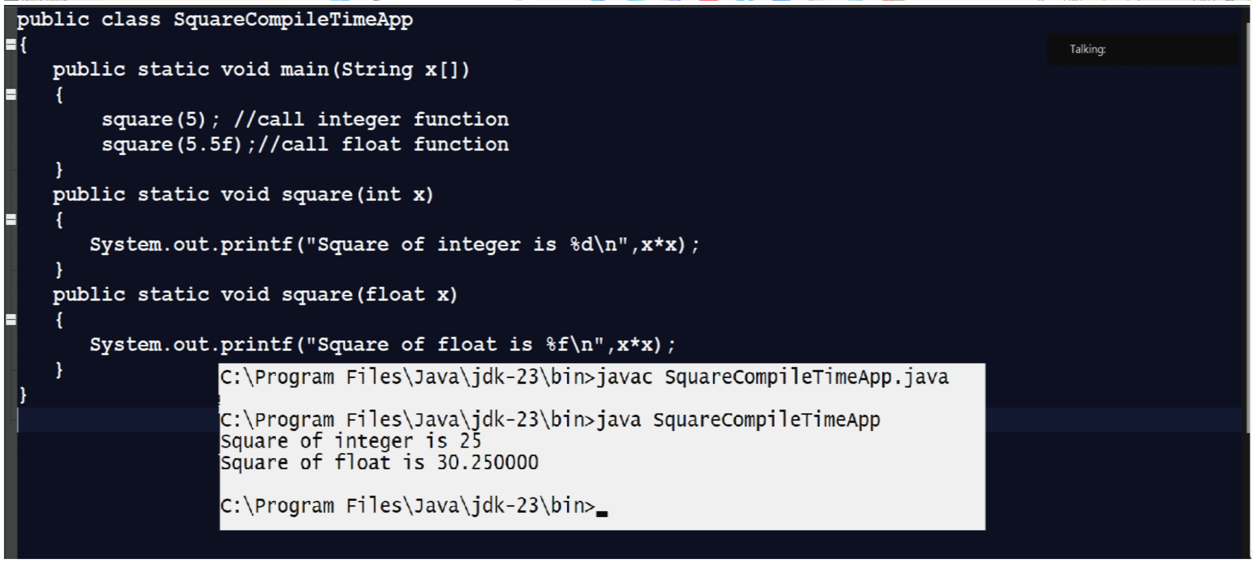
If we want to work with overloading we have some important points.

1. Function name must be same
2. Function return type is not consider means you can give different return type to every function
  Definition if required.
3. Function parameter type or parameter list or sequence of parameter in every definition must be different
4. Function calling is dependent on number of parameter pass in it, its sequence and its data type.



```
public class SquareCompileTimeApp
{
   public static void main(String x[ ])
   {
          square(5);  //call integer function
          square(5.5f);  // call float function
   }
   public static void square(int x) //definition or logical block of function
   {
       System.out.printf("Square of integer is %d\n",x*x);
   }
   public static void square(float x) //definition
   {
       System.out.printf("Square of float is %f\n",x*x);
   }
}
```

1. Function name must be same
2. Function return type is not consider means you can give different return type to every function
  Definition if required.
3. Function parameter type or parameter list or sequence of parameter in every definition must be different
4. Function calling is dependent on number of parameter pass in it, its sequence and its data type.

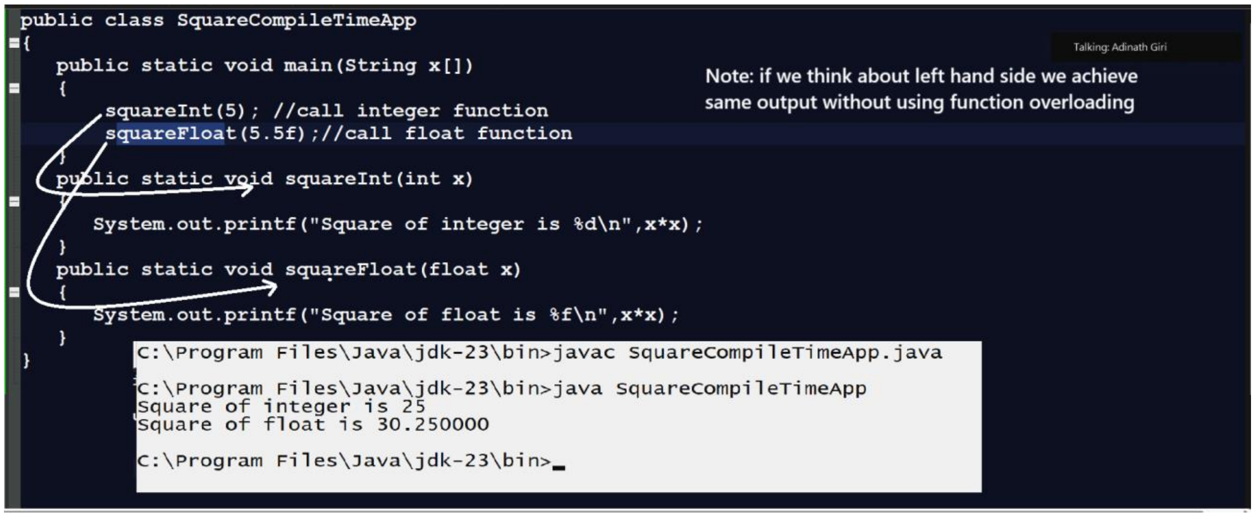


```
public class SquareCompileTimeApp
{
   public static void main(String x[])
   {
       square(5); //call integer function
       square(5.5f);//call float function
   }
   public static void square(int x)
   {
      System.out.printf("Square of integer is %d\n",x*x);
   }
   public static void square(float x)
   {
      System.out.printf("Square of float is %f\n",x*x);
   }
}
```

```
C:\Program Files\Java\jdk-23\bin>javac SquareCompileTimeApp.java

C:\Program Files\Java\jdk-23\bin>java SquareCompileTimeApp
Square of integer is 25
Square of float is 30.250000

C:\Program Files\Java\jdk-23\bin>
```



```
public class SquareCompileTimeApp
{
   public static void main(String x[])
   {
      squareInt(5); //call integer function
      squareFloat(5.5f);//call float function
   }
   public static void squareInt(int x)
   {
      System.out.printf("Square of integer is %d\n",x*x);
   }
   public static void squareFloat(float x)
   {
      System.out.printf("Square of float is %f\n",x*x);
   }
}
```

Talking: Adinath Giri

Note: if we think about left hand side we achieve same output without using function overloading

```
C:\Program Files\Java\jdk-23\bin>javac SquareCompileTimeApp.java

C:\Program Files\Java\jdk-23\bin>java SquareCompileTimeApp
Square of integer is 25
Square of float is 30.250000

C:\Program Files\Java\jdk-23\bin>
```

Q. Why use function overloading or what is benefit of function overloading?

____

Sometime we have different logics under the same domain and every logic required different parameter list or may be data then writing different function name to every function create burden on developer to remember the multiple function names at the time of function use or calling so better way you can give same name to all functions so the benefit is developer not need to remember multiple function names this is the benefit of function overloading

If we think about real time API they use the same function name for different purpose to reduce the developer load.