Constructor

Click to Continue

Constructor is a method used for creating an object of a class and can be used to initialize member variables of the class.

Salient points:

- *Constructors* and name of the class should be same.
- Constructor should not return any value, not even void.
- Constructor are not mandatory.
- If no constructor defined java will use the default constructor to create objects.

Default Constructor

Click to Continue

Default constructor are constructor with no parameters If the class does not define any constructors, default constructor is automatically invoked by the JVM to create objects.

```
public class Employee {

    public Employee() {
        // default constructor
        // member variable initialization can be done here
    }

}
```

Default Constructor

Overloading Constructors

Click to Continue

Overloaded constructors are nothing but default constructor with arguments.

Step 1: Create a default constructor.

Step 2: Add the desired arguments for the constructor (method) to create an overloaded constructor.

Any number of constructors can be overloaded.

Try it Out– Let us overload constructor?

Click to Continue

Create a Employee object with members age, name and overload the constructor using the fields.

```
public class Employee {

    int age =0;
    String name;

    public Employee() {
        // default constructor
        // member variable initialization can be done here
    }

    public Employee(int age) {
        this.age = age;
    }

    public Employee(String name) {
        super();
        this.name = name;
    }

    public Employee(int age, String name) {
        this.age = age;
        this.name = name;
    }

}
```

Default Constructor

Overloaded Constructors

“this” Keyword

Click to Continue

“**this**” keyword refer to the current object instance as below,

- It can be used to refer instance variables and not for static or class variables
- It can also be used to invoke the overloaded constructors.
- Used in mutators to set the value of the instance variable with the setter variable argument.

“this” Keyword

Click to Continue

“**this**” keyword refer to the current object instance as below,

- It can be used to refer instance variables and not for static or class variables
- It can also be used to invoke the overloaded constructors.
- Used in mutators to set the value of the instance variable with the se

argument.

"this" Keyword

Click to Continue

"**this**" keyword refer to the current object instance as below,

- It can be used to refer instance variables and not for static or class variables
- It can also be used to invoke the overloaded constructors.
- Used in mutators to set the value of the instance variable with the se
argument.

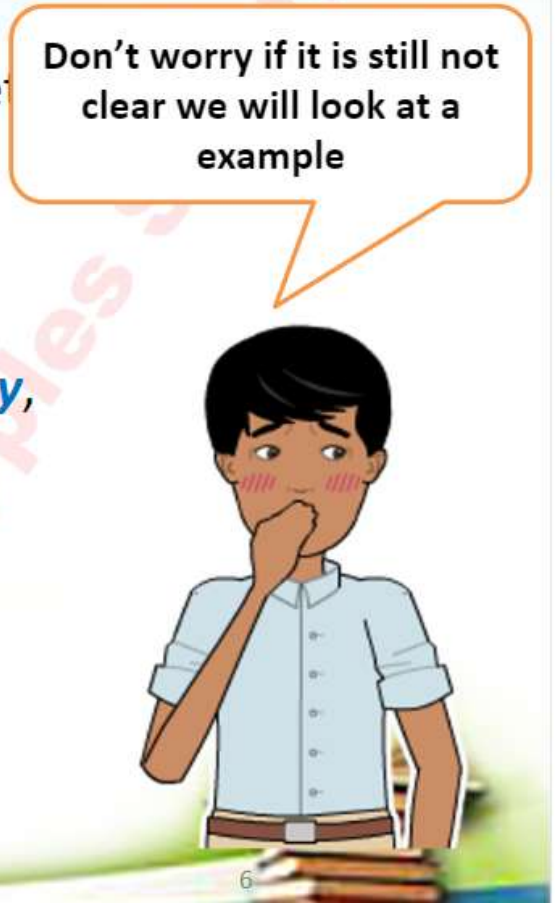

Illustration: A method ***calculateSalary*** has a parameter ***basicSalary***, also the class has the instance variable ***basicSalary***. The following statement refers to the member variable though the method argument shadows the member variable.

```
this.basicSalary;
```

“this” Reference – illustration

Click to Continue

```
public class Employee {

    int age =0;
    String name;

    public Employee() {
        // default constructor
        // member variable initialization can be done here
    }

    public Employee(int age) {
        this.age = age;
    }

    public Employee(String name) {
        super();
        this.name = name;
    }

    public Employee(int age, String name) {
        this.age = age;
        this.name = name;
    }

}
```

“this” keyword used to refer to the member variables.

Chaining Constructor

Click to Continue

Constructor calls can be *chained*, which means one can invoke a constructor from another constructor from the same class.

Syntax: "this()" is used for invoking constructors from other constructor of a class.

Points to remember:

- "this()" constructor call must occur as the first statement in a constructor.
- "this()" call can then be followed by any other statements.

Try it out - this()

Click to Continue

```
public class Employee {

    int age =0;
    String name;

    public Employee() {
        // default constructor
        // member variable initialization can be done here
    }

    public Employee(int age) {
        this.age = age;
    }

    public Employee(String name) {
        this.name = name;
    }

    public Employee(int age, String name) {
        this(name);
        this.age = age;
        this.name = name;
    }

}
```

This overloaded constructor is chained with the Employee(String) constructor.