

Constructor

Q. What is constructor?

Constructor is a function same name as class name but without return type called as constructor

Syntax of constructor

```
class classname
{
    classname() //constructor
    {
    }
}
```

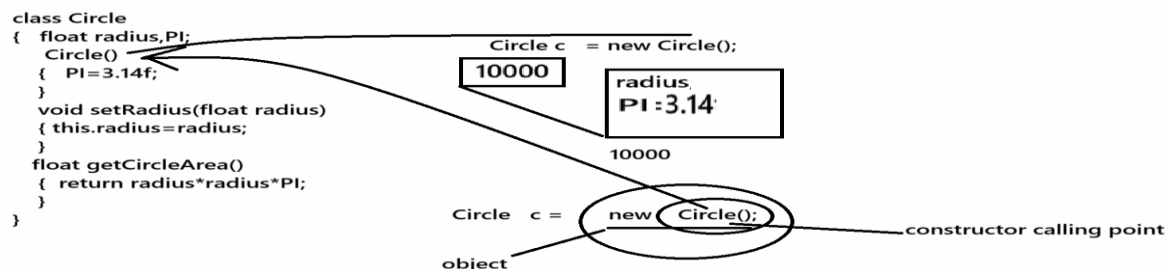
Example:

```
class ABC{
    ABC(){
    }
}
```

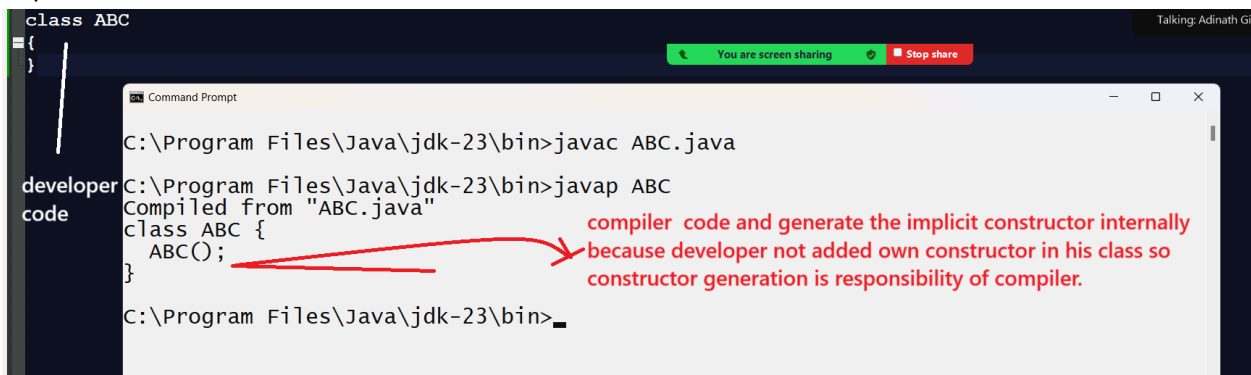
Q. Why use constructor or what is benefit of constructor?

1. Initialize the class member at the time of object creation or object initialization purpose

2. To call function at the time of object creation.



Note: if we declare not constructor within class then every class has one default constructor known as implicit constructor.



If we want to work with constructor we have some types of constructor

1. Default constructor: default constructor means user defined no argumented constructor called as default constructor

```
class ABC
{
    ABC() ——— Default constructor
    {
    }
}
```

Note: Normally default constructor is choice of developer as per coding to initialize fix values at the time of object creation in instance variable or static variable

```
class Circle
{ float radius,PI;
  Circle()
  {
    PI=3.14f;
  }
  void setRadius(float radius)
  { this.radius=radius;
  }
  float getArea(){
    return radius*radius*PI;
  }
}

public class CircleApp
{
  public static void main(String x[])
  {
    Circle c = new Circle();
    c.setRadius(3.0f);
    float result=c.getArea();
    System.out.printf("Area of circle is %f\n",result);
  }
}

or

import java.util.*;
class Fibo
{
  int f1,f2,fib;
  Fibo(){
    f1=0;
```

```

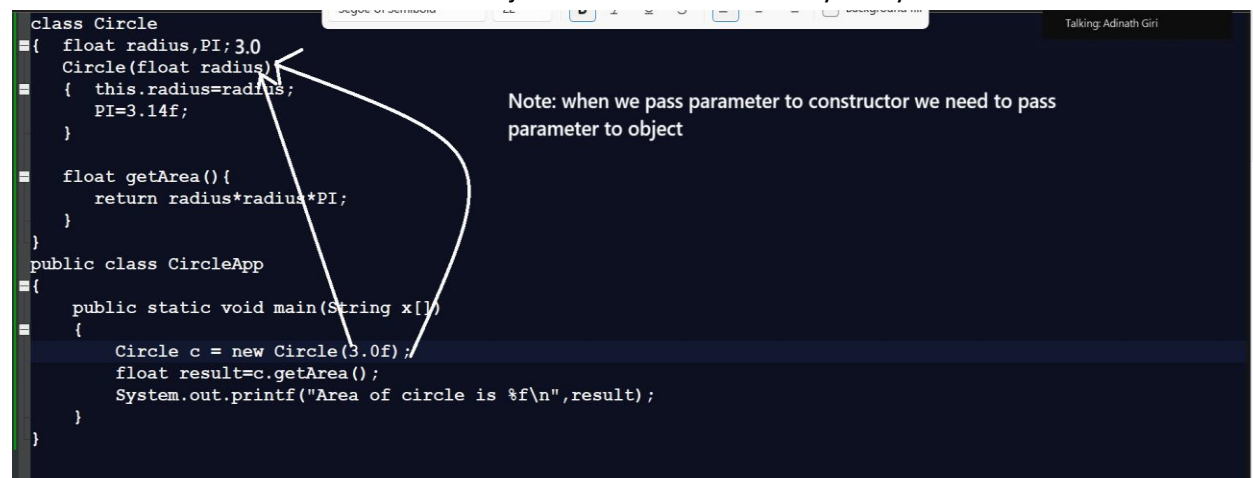
        f2=1;
    }
    void printSeries(int limit){
        System.out.printf("%d\n%d\n",f1,f2);
        for(int i=1;i<=limit;i++)
        { fib=f1+f2;
          f1=f2;
          f2=fib;
          System.out.printf("%d\n",fib);
        }
    }
}
}
public class FiboApp
{ public static void main(String x[])
{
    Fibo f1 = new Fibo();
    Scanner xyz = new Scanner(System.in);
    System.out.println("Enter number");
    int no=xyz.nextInt();
    f1.printSeries(no);
}
}

```

2. parameterized constructor or argumented constructor

When we pass parameter to constructor called as parameterized constructor.

Normally Parameterized constructor recommended by developer we want to initialize value to instance variable or class variable at the time of object creation but value vary every time or not fixed



```

class Circle
{ float radius,PI;3.0
  Circle(float radius)
  { this.radius=radius;
    PI=3.14f;
  }

  float getArea(){
    return radius*radius*PI;
  }
}

public class CircleApp
{
  public static void main(String x[])
  {
    Circle c = new Circle(3.0f);
    float result=c.getArea();
    System.out.printf("Area of circle is %f\n",result);
  }
}

```

Note: when we pass parameter to constructor we need to pass parameter to object

3. Overloaded constructor

Constructor overloading is part of compile time polymorphism means we define multiple constructors with different parameter list or with a different parameter type or different parameter sequence then we can use constructor overloading.

Note: if we think about constructor overloading which constructor should execute is dependent on number of parameter pass in it , its sequence and its data type.

```
class Area
{ float radius; 3.0
  Area(float radius)
  { this.radius=radius;
  }
  Area(int len,int wid)
  { this.len=len;
    this.wid=wid;
  }
  float getCircleArea(){
    return radius*radius*3.14f;
  }
  float getRectArea(){
    return len*width;
  }
}

public class AreaApp
{
  public static void main(String x[])
  {
    Area a1 = new Area(5,4); // two parameter constructor
    Area a2 = new Area(3.0f);

    float circleArea=a1.getCircleArea();
    System.out.printf("Circle area is %f\n",circleArea);

    float rectArea=a1.getRectArea();
    System.out.printf("\nRectangle area is %d\n",rectArea);
  }
}
```

Source code with example

```
class Area
{ float radius,PI=3.14f;
  int len,wid;
  Area(float radius)
  { this.radius=radius;
  }
  Area(int len,int wid)
  { this.len=len;
    this.wid=wid;
  }
  float getCircleArea()
  { return radius*radius*PI;
  }
  int getRectArea()
  { return len*wid;
  }
}

public class AreaApplication
```

```

{ public static void main(String x[])
{
    Area a1 = new Area(5,4);
    int rectArea=a1.getRectArea();
    System.out.printf("\nRectangle area is %d\n",rectArea);
    Area a2 = new Area(3.0f);
    float cirArea=a2.getCircleArea();
    System.out.printf("\nCircle area is %f\n",cirArea);
}
}

```

Example: Suppose consider we have class name as Sort with two constructors

Sort(int a[]): this constructor accept integer array as parameter

Sort(char ch[]): this constructor accept character array as parameter.

int[] getIntSortedArray(): this function can return the integer sorted array

char [] getCharSortedArray(): this function can return character sorted array

```

import java.util.*;
class Sort
{
    int a[];
    char ch[];
    Sort(int a[])
    { this.a=a;
    }
    Sort(char ch[])
    { this.ch=ch;
    }
    int [] getIntSortedArray(){
        for(int i=0; i<a.length; i++)
        {
            for(int j=(i+1); j<a.length; j++)
            {
                if(a[i]>a[j])
                {
                    int temp=a[i];
                    a[i]=a[j];
                    a[j]=temp;
                }
            }
        }
        return a;
    }
}

```

```

    }
    char [] getCharSortedArray()
    { for(int i=0; i<ch.length; i++)
        {
            for(int j=(i+1); j<ch.length; j++)
            {
                if(ch[i]>ch[j])
                {
                    char temp=ch[i];
                    ch[i]=ch[j];
                    ch[j]=temp;
                }
            }
        }
        return ch;
    }
}

public class SortingApplication
{
    public static void main(String x[])
    { Scanner xyz = new Scanner(System.in);
        int a[]=new int[5];
        char ch[]=new char[5];
        System.out.println("Enter data in integer array");
        for(int i=0;i<a.length;i++)
        { a[i]=xyz.nextInt();
        }
        Sort s = new Sort(a); //call integer array constructor
        System.out.println("Enter data in character array");
        xyz.nextLine();
        for(int i=0; i<ch.length;i++)
        {
            ch[i]=xyz.nextLine().charAt(0);
        }

        Sort s1 = new Sort(ch); //call character array constructor

        int result[]=s.getIntSortedArray();
        char res []=s1.getCharSortedArray();
        System.out.println("Display integer sorted array");
        for(int i=0;i<result.length;i++)
        { System.out.printf("%d\t",result[i]);

```

```

    }
    System.out.println("\nDisplay character sorted array");
    for(int i=0;i<res.length;i++)
    { System.out.printf("%c\t",res[i]);
    }
    }
}

```

Example: Suppose consider we are working for Hiring Agency and they hire the fresher as well as experience people so you can manage it by overloading like as

```

class Experience
{ private String name;
  private String email;
  private String contact;
  private String prevComp;
  private int expSal;
  private int prevSal;

  public void setName(String name)
  { this.name=name;
  }
  public String getName(){
    return name;
  }
  public void setEmail(String email)
  { this.email=email;
  }
  public String getEmail(){
    return email;
  }
  public void setContact(String contact)
  { this.contact=contact;
  }
  public String getContact(){
    return contact;
  }
  public void setPrevComp(String prevComp)
  { this.prevComp=prevComp;
  }
  public String getPrevComp(){
    return prevComp;
  }
}

```

```

    }
    public void setPrevSal(int prevSal)
    { this.prevSal=prevSal;
    }
    public int getPrevSal(){
        return prevSal;
    }
    public void setExpSal(int expSal)
    { this.expSal=expSal;
    }
    public int getExpSal(){
        return expSal;
    }
}

class Fresher
{ private String name;
  private String email;
  private String contact;
  private String degreeCert;
  private int expSal;

  public void setName(String name)
  { this.name=name;
  }
  public String getName(){
      return name;
  }
  public void setEmail(String email)
  { this.email=email;
  }
  public String getEmail(){
      return email;
  }
  public void setContact(String contact)
  { this.contact=contact;
  }
  public String getContact(){
      return contact;
  }
  public void setDegreeCert(String degreeCert)
  { this.degreeCert=degreeCert;
  }
}

```



```

    public String getDegreeCert(){
        return degreeCert;
    }
    public void setExpSal(int expSal)
        { this.expSal=expSal;
        }
        public int getExpSal(){
            return expSal;
        }
    }
class Recruiter
{ private Experience e;
  private Fresher f;
  Recruiter(Experience e)
  { this.e=e;
  }
  Recruiter(Fresher f)
  {this.f=f;
  }
  void showExperience()
  {
      String name=e.getName();
      String email=e.getEmail();
      String contact=e.getContact();
      String prevComp=e.getPrevComp();
      int prevSal=e.getPrevSal();
      int expSal=e.getExpSal();
      System.out.println("Experience Details\n");
      System.out.println("=====");
      System.out.println("Name is "+name);
      System.out.println("Email is "+email);
      System.out.println("Contact is "+contact);
      System.out.println("Previous Company "+prevComp);
      System.out.println("Previous Salary is "+prevSal);
      System.out.println("Expected Salary is "+expSal);
  }
  void showFresher()
  { String name=f.getName();
    String email=f.getEmail();
    String contact=f.getContact();
    String degreeCert=f.getDegreeCert();
    int expSal=f.getExpSal();

```

```

        System.out.println("=====");
        System.out.println("Frehser Details Details\n");
        System.out.println("=====");
        System.out.println("Name is "+name);
        System.out.println("Email is "+email);
        System.out.println("Contact is "+contact);
        System.out.println("Degree Certificate "+degreeCert);
        System.out.println("Expected Salary is "+expSal);
    }

}

public class RecruiterApp
{
    public static void main(String x[])
    {
        Experience e = new Experience();
        e.setName("RAM");
        e.setEmail("ram@gmail.com");
        e.setContact("1234567891");
        e.setPrevComp("TCS");
        e.setPrevSal(100000);
        e.setExpSal(150000);

        Fresher f=new Fresher();
        f.setName("Shyam");
        f.setEmail("shyam@gmail.com");
        f.setContact("2233445566");
        f.setDegreeCert("BTECH");
        f.setExpSal(30000);

        Recruiter r1 = new Recruiter(e);
        Recruiter r2 = new Recruiter(f);
        r1.showExperience();
        r2.showFresher();

    }
}

```

or

Example with source code

```

class Experience
{ private String name;

```

```
private String email;
private String contact;
private String prevComp;
private int expSal;
private int prevSal;

public Experience(String name,String email,String contact,String prevComp,int prevSal,int expSal){
    this.name=name;
    this.email=email;
    this.contact=contact;
    this.prevComp=prevComp;
    this.expSal=expSal;
    this.prevSal=prevSal;
}

public void setName(String name)
{ this.name=name;
}
public String getName(){
    return name;
}
public void setEmail(String email)
{ this.email=email;
}
public String getEmail(){
    return email;
}
public void setContact(String contact)
    { this.contact=contact;
    }
public String getContact(){
    return contact;
}
public void setPrevComp(String prevComp)
{ this.prevComp=prevComp;
}
public String getPrevComp(){
    return prevComp;
}
public void setPrevSal(int prevSal)
{ this.prevSal=prevSal;
}
```

```

    public int getPrevSal(){
        return prevSal;
    }
    public void setExpSal(int expSal)
    { this.expSal=expSal;
    }
    public int getExpSal(){
        return expSal;
    }
}

class Fresher
{ private String name;
  private String email;
  private String contact;
  private String degreeCert;
  private int expSal;

  Fresher(){
  }

  Fresher(String name,String email,String contact,String degreeCert,int expSal)
  { this.name=name;
    this.email=email;
    this.contact=contact;
    this.degreeCert=degreeCert;
    this.expSal=expSal;
  }
  public void setName(String name)
  { this.name=name;
  }
  public String getName(){
    return name;
  }
  public void setEmail(String email)
  { this.email=email;
  }
  public String getEmail(){
    return email;
  }
  public void setContact(String contact)
  { this.contact=contact;
  }
}

```

```

public String getContact(){
    return contact;
}
public void setDegreeCert(String degreeCert)
{ this.degreeCert=degreeCert;
}
public String getDegreeCert(){
    return degreeCert;
}
public void setExpSal(int expSal)
    { this.expSal=expSal;
    }
    public int getExpSal(){
        return expSal;
    }
}

class Recruiter
{ private Experience e;
  private Fresher f;
  Recruiter(Experience e)
  { this.e=e;
  }
  Recruiter(Fresher f)
  {this.f=f;
  }
  void showExperience()
  {
      String name=e.getName();
      String email=e.getEmail();
      String contact=e.getContact();
      String prevComp=e.getPrevComp();
      int prevSal=e.getPrevSal();
      int expSal=e.getExpSal();
      System.out.println("Experience Details\n");
      System.out.println("=====");
      System.out.println("Name is "+name);
      System.out.println("Email is "+email);
      System.out.println("Contact is "+contact);
      System.out.println("Previous Company "+prevComp);
      System.out.println("Previous Salary is "+prevSal);
      System.out.println("Expected Salary is "+expSal);
  }
}

```

```

void showFresher()
{ String name=f.getName();
  String email=f.getEmail();
  String contact=f.getContact();
  String degreeCert=f.getDegreeCert();
  int expSal=f.getExpSal();
  System.out.println("=====");
  System.out.println("Frehser Details Details\n");
  System.out.println("=====");
  System.out.println("Name is "+name);
  System.out.println("Email is "+email);
  System.out.println("Contact is "+contact);
  System.out.println("Degree Certificate "+degreeCert);
  System.out.println("Expected Salary is "+expSal);
}

}

public class RecruiterApp
{
  public static void main(String x[])
  {
    Experience e = new Experience("RAM","ram@gmail.com","1234567891","TCS",100000,150000);

    Fresher f=new Fresher("Shyam","shyam@gmail.com","2233445566","BTECH",30000);

    Recruiter r1 = new Recruiter(e);
    Recruiter r2 = new Recruiter(f);
    r1.showExperience();
    r2.showFresher();

  }
}
or

```

Example with source code

```

class Experience
{ private String name;
  private String email;
  private String contact;
  private String prevComp;
  private int expSal;
  private int prevSal;

```

```
public Experience(String name,String email,String contact,String prevComp,int prevSal,int expSal){  
    this.name=name;  
    this.email=email;  
    this.contact=contact;  
    this.prevComp=prevComp;  
    this.expSal=expSal;  
    this.prevSal=prevSal;  
}
```

```
public void setName(String name)  
{ this.name=name;  
}  
public String getName(){  
    return name;  
}  
public void setEmail(String email)  
{ this.email=email;  
}  
public String getEmail(){  
    return email;  
}  
public void setContact(String contact)  
    { this.contact=contact;  
    }  
public String getContact(){  
    return contact;  
}  
public void setPrevComp(String prevComp)  
{ this.prevComp=prevComp;  
}  
public String getPrevComp(){  
    return prevComp;  
}  
public void setPrevSal(int prevSal)  
{ this.prevSal=prevSal;  
}  
public int getPrevSal(){  
    return prevSal;  
}  
    public void setExpSal(int expSal)  
    { this.expSal=expSal;
```

```

    }
    public int getExpSal(){
        return expSal;
    }
}
class Fresher
{ private String name;
  private String email;
  private String contact;
  private String degreeCert;
  private int expSal;

  Fresher(){
  }

  Fresher(String name,String email,String contact,String degreeCert,int expSal)
  { this.name=name;
    this.email=email;
    this.contact=contact;
    this.degreeCert=degreeCert;
    this.expSal=expSal;
  }
  public void setName(String name)
  { this.name=name;
  }
  public String getName(){
    return name;
  }
  public void setEmail(String email)
  { this.email=email;
  }
  public String getEmail(){
    return email;
  }
  public void setContact(String contact)
  { this.contact=contact;
  }
  public String getContact(){
    return contact;
  }
  public void setDegreeCert(String degreeCert)
  { this.degreeCert=degreeCert;

```



```

    }
    public String getDegreeCert(){
        return degreeCert;
    }
    public void setExpSal(int expSal)
        { this.expSal=expSal;
        }
        public int getExpSal(){
            return expSal;
        }
    }
class Recruiter
{ private Experience e;
  private Fresher f;
  Recruiter(Experience e)
  { this.e=e;
  }
  Recruiter(Fresher f)
  {this.f=f;
  }
  void showExperience()
  {
      String name=e.getName();
      String email=e.getEmail();
      String contact=e.getContact();
      String prevComp=e.getPrevComp();
      int prevSal=e.getPrevSal();
      int expSal=e.getExpSal();
      System.out.println("Experience Details\n");
      System.out.println("=====");
      System.out.println("Name is "+name);
      System.out.println("Email is "+email);
      System.out.println("Contact is "+contact);
      System.out.println("Previous Company "+prevComp);
      System.out.println("Previous Salary is "+prevSal);
      System.out.println("Expected Salary is "+expSal);
  }
  void showFresher()
  { String name=f.getName();
    String email=f.getEmail();
    String contact=f.getContact();
    String degreeCert=f.getDegreeCert();

```

```

        int expSal=f.getExpSal();
        System.out.println("=====");
        System.out.println("Frehser Details Details\n");
        System.out.println("=====");
        System.out.println("Name is "+name);
        System.out.println("Email is "+email);
        System.out.println("Contact is "+contact);
        System.out.println("Degree Certificate "+degreeCert);
        System.out.println("Expected Salary is "+expSal);
    }

}

public class RecruiterApp
{
    public static void main(String x[])
    {

        Recruiter r1 = new Recruiter(new
Experience("RAM","ram@gmail.com","1234567891","TCS",100000,150000));
        Recruiter r2 = new Recruiter(new
Fresher("Shyam","shyam@gmail.com","2233445566","BTECH",30000));
        r1.showExperience();
        r2.showFresher();

    }
}

or
class Experience
{ private String name;
    private String email;
    private String contact;
    private String prevComp;
    private int expSal;
    private int prevSal;

    public Experience(String name,String email,String contact,String prevComp,int prevSal,int expSal){
        this.name=name;
        this.email=email;
        this.contact=contact;
        this.prevComp=prevComp;
    }
}

```

```

        this.expSal=expSal;
        this.prevSal=prevSal;
    }

    public void setName(String name)
    { this.name=name;
    }
    public String getName(){
        return name;
    }
    public void setEmail(String email)
    { this.email=email;
    }
    public String getEmail(){
        return email;
    }
    public void setContact(String contact)
        { this.contact=contact;
        }
    public String getContact(){
        return contact;
    }
    public void setPrevComp(String prevComp)
    { this.prevComp=prevComp;
    }
    public String getPrevComp(){
        return prevComp;
    }
    public void setPrevSal(int prevSal)
    { this.prevSal=prevSal;
    }
    public int getPrevSal(){
        return prevSal;
        }
        public void setExpSal(int expSal)
        { this.expSal=expSal;
        }
        public int getExpSal(){
            return expSal;
        }
    }
}
class Fresher

```

```

{ private String name;
  private String email;
  private String contact;
  private String degreeCert;
  private int expSal;

  Fresher(){
  }

  Fresher(String name,String email,String contact,String degreeCert,int expSal)
  { this.name=name;
    this.email=email;
      this.contact=contact;
      this.degreeCert=degreeCert;
      this.expSal=expSal;
  }
  public void setName(String name)
  { this.name=name;
  }
  public String getName(){
    return name;
  }
  public void setEmail(String email)
  { this.email=email;
  }
  public String getEmail(){
    return email;
  }
  public void setContact(String contact)
    { this.contact=contact;
    }
  public String getContact(){
    return contact;
  }
  public void setDegreeCert(String degreeCert)
  { this.degreeCert=degreeCert;
  }
  public String getDegreeCert(){
    return degreeCert;
  }
  public void setExpSal(int expSal)
    { this.expSal=expSal;

```

```

    }
    public int getExpSal(){
        return expSal;
    }
}

class Recruiter
{ private Experience e;
  private Fresher f;
  Recruiter(Experience e)
  { this.e=e;
  }
  Recruiter(Fresher f)
  {this.f=f;
  }
  void showExperience()
  {
      String name=e.getName();
      String email=e.getEmail();
      String contact=e.getContact();
      String prevComp=e.getPrevComp();
      int prevSal=e.getPrevSal();
      int expSal=e.getExpSal();
      System.out.println("Experience Details\n");
      System.out.println("=====");
      System.out.println("Name is "+name);
      System.out.println("Email is "+email);
      System.out.println("Contact is "+contact);
      System.out.println("Previous Company "+prevComp);
      System.out.println("Previous Salary is "+prevSal);
      System.out.println("Expected Salary is "+expSal);
  }
  void showFresher()
  {
      String name=f.getName();
      String email=f.getEmail();
      String contact=f.getContact();
      String degreeCert=f.getDegreeCert();
      int expSal=f.getExpSal();
      System.out.println("=====");
      System.out.println("Fresher Details\n");
      System.out.println("=====");
      System.out.println("Name is "+name);
      System.out.println("Email is "+email);
  }
}

```

```

        System.out.println("Contact is "+contact);
        System.out.println("Degree Certificate "+degreeCert);
        System.out.println("Expected Salary is "+expSal);
    }

}

public class RecruiterApp
{
    public static void main(String x[])
    {
        new Recruiter(new
Experience("RAM","ram@gmail.com","1234567891","TCS",100000,150000)).showExperience();
        new Recruiter(new
Fresher("Shyam","shyam@gmail.com","2233445566","BTECH",30000)).showFresher();

    }
}

```

4. this() constructor

this() constructor is used for perform constructor chaining

Q. What is constructor chaining in JAVA?

Constructor chaining is a concept where one constructor can call from another constructor without creating object by using new keyword.

If we want to perform constructor chaining in java we have two ways.

- 1. Using same class:** if we want to perform constructor within a same class then we can use constructor overloading concept with this () constructor
- 2. Using inheritance:** if we want to perform constructor chaining by using inheritance we can use super() constructor

Note: super () constructor we will discuss in next chapter i.e in inheritance.

Now we want to perform constructor chaining using this() constructor

If we want to perform constructor chaining using this () constructor we need to use the constructor overloading concept.

```

class A
{
    A()
    {
        this(5);
        System.out.println("I am default constructor");
    }
    A(int x)
    {
        this(5.5f);
        System.out.println("I am integer param constructor "+x);
    }
    A(float x)
    {
        System.out.println("I am float param constructor "+x);
    }
}

public class ConsChainApp
{
    public static void main(String x[])
    {
        A a1 = new A(); //default constructor
    }
}

```

Talking: [redacted]

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

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

```

I am float param constructor5.5
I am integer param constructor 5
I am default constructor

```

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

Note: if we think about this() constructor then this() construct must be first line of code in child class constructor.

If we want to work with constructor we need to know the some important points related with constructor

1. Constructor cannot have return type

If you give return type to constructor then constructor consider as method in java means we can say in java function name and class name may be same.

```

class A
{
    void A()
    {
        System.out.println("Hello");
    }
}

public class ConsChainApp
{
    public static void main(String x[])
    {
        A a1 = new A();
        a1.A();
    }
}

```

consider as function
so we need to call it
using object.

```

C:\Program Files\Java\jdk-23\bin>javac ConsChainApp.j
C:\Program Files\Java\jdk-23\bin>java ConsChainApp
Hello
C:\Program Files\Java\jdk-23\bin>

```

2) Cannot declare constructor as static

If we declare the constructor as static then we get compile time error because constructor call when we create object of class and static member allocate memory before object means we can say constructor calling and static keyword working principle opposite of each other so we cannot declare constructor as static and if we declare it as static then we get compile time error.

```
class A
{
    static A()
    { System.out.println("Hello");
    }
}

public class ConsChainApp
{
    public static void main(String x[])
    {
        A a1 = new A();
    }
}
```

C:\Program Files\Java\jdk-23\bin>java ConsChainApp

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

ConsChainApp.java:3: error: modifier static not allowed here

static A()

^

1 error

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

3) Try to avoid declare constructor as private.

When we declare the constructor as private then we cannot create class object outside of his class definition

```
class A
{
    private A()
    { System.out.println("Hello");
    }
}

public class ConsChainApp
{
    public static void main(String x[])
    {
        A a1 = new A();
    }
}
```

Note: if we think about left hand side code we get compile time error because we declare our constructor as private and try to create object of class outside of class definition and it is not possible so we get compile time error.

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

ConsChainApp.java:9: error: A() has private access in A

{ A a1 = new A();

^

1 error

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

Note: if we want to create single tone class or utility class in java then class constructor must be private.

Q. What is utility class?

Utility class means class which contain private constructor and static methods called as utility class means we can say we cannot create object of utility class and we can use its static method without object by using class name

Example of utility class?

```
class A
{
    private A()
    {
    }
    static void show(){
        System.out.println("I am show method in utility class");
    }
    static void display()
    { System.out.println("I am display method in utility class");
    }
}

public class ConsChainApp
{
    public static void main(String x[])
    {
        A.show();
        A.display();
    }
}
```

```
C:\Program Files\Java\jdk-23\bin>javac ConsChainApp.java
C:\Program Files\Java\jdk-23\bin>java ConsChainApp
I am show method in utility class
I am display method in utility class
C:\Program Files\Java\jdk-23\bin>
```

Q. What is singleton class?

Singleton class is a design pattern and if we use the singleton design pattern then we can create only one object of class in whole application.

Q. What is design pattern?

Design pattern are the some standard principle in software development which help us to develop the better code and manage the some limitation of OOP.

How to implement singleton design pattern practically

Steps to work with Singleton design pattern

1. Declare class with private constructor

```
class Test
{
    private Test()
    {
        System.out.println("I am singleton class");
    }
}
```

2. Declare the same class reference in it with private static

```
class Test
{ private static Test t=null;
```

```

private Test()
{
    System.out.println("I am singleton class");
}
}

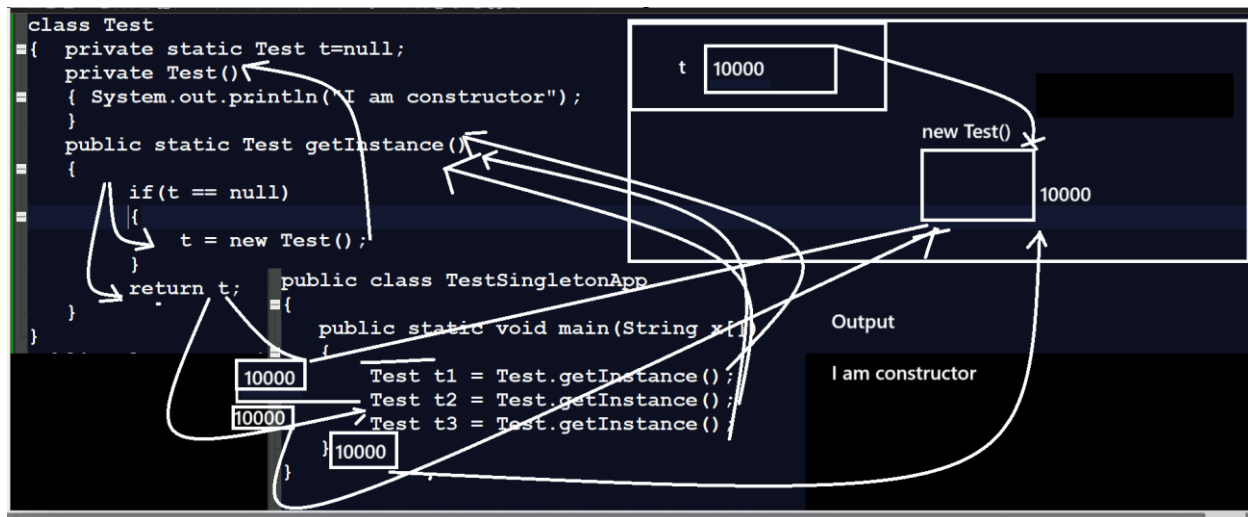
```

3. Define static method which returns reference of same class with following logics

```

class Test
{ private static Test t=null;
  private Test()
  { System.out.println("I am singleton class");
  }
  public static Test getTest()
  {
      if(t==null)
      {
          t = new Test();
      }
      return t;
  }
}

```



Q. Can we declare constructor as final?

No

Note: we will discuss reason in inheritance chapter.

Q. Can we declare constructor as abstract?

No

Note: we will discuss reason in inheritance chapter

Q. What is difference between function and constructor?

Function	Constructor
Function can have return type	Constructor cannot have return type
Function need to call for execute its logic using object.functionname or reference.functionname	Constructor not need to call manually it is call automatically at the time of object creation
There is no implicit function in class	But there is implicit constructor in class means when user not provide constructor to class then class define own constructor known as implicit constructor
Function support to recursion	Constructor not support to recursion
Function can declare as static	Constructor cannot declare as static
Function can declare as final	Constructor cannot declare as final
Function can declare as abstract	Constructor cannot declare as abstract
According to coding standard function is used for write the logics	According coding standard constructor is used for initialize class variable at the time of object creation
Function can implement dynamic polymorphism with the help of overriding	Constructor cannot implement dynamic polymorphism with the help of overriding.