```
while(true)
{
  System.out.println("Do you want to contine [Yes/no]:");
  String choice = scanner.next();  //NO  OR No

   if(choice.equals("no"))
  {
      break;
  }

}
```

## Program on equals(), equalsIgnoreCase() and isEmpty() method :

```java
package com.ravi;

import java.util.Scanner;

public class StringComparison {

    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your first Name :");
        String fname = sc.next();

        if(fname.equalsIgnoreCase("James"))  //equals()   equalsIgnoreCase()
        {
            System.out.println("Your name is :"+fname);
        }
        else
        {
            System.err.println("You are not James!!!");
        }


        System.out.println("..............");

        String str = "Hello";
        System.out.println("Is String Empty ?"+str.isEmpty());

        sc.close();
    }
}
```

## Method return type as a class :

- In java, We cannot define a method without return type.
- As a method return type, We have so many possibilities are available :
  a) Method return type as  a void.
  b) Method return type as a primitive data type.
  c) Method return type as a **class**/ interface/ enum/ record as so on.

### Method return type Example :

```
public int accept()
{
  return 5.5;
}
```

The return value (5.5) of the method must be compaitable with return type (int) of the method. In the example it is not compaitable so we will get an error

### Example :

```
public class Test
{
   public Test accept()
   {

      return new Test();
   }
}
```

If we are able to assign the return value of the method (new Test()) to the return type (Test) of the method using a variable then it is compaitable

Test t = new Test();

return type of the method

return value of the method

### Programs :

```java
package com.ravi;

public class Test
{
    public Test accept()
    {
        return new Test();
    }

}
```

Note : The return value is calling the default constructor (added by javac) of Test class

### //Program :

```java
package com.ravi;

public class Demo
{
    int x;

    public Demo(int x)
    {
        this.x = x;
    }

    public Demo accept()
    {
        return new Demo(15);
    }

}
```

Note : In order to return the value we depend upon parameterized constructor

## What is a Factory Method ?

- If a method return type is class OR interface that means from that particular method we can return the object then it is called Factory method.

//Program

```java
package com.ravi.factory_method;

public class Book
{
    private String title;
    private String author;

    public Book(String title, String author)
    {
        super();
        this.title = title;
        this.author = author;
    }

    @Override
    public String toString()
    {
        return "Book [title=" + title + ", author=" + author + "]";
    }

    public static Book getBookObject()  //static factory method
    {
        return new Book("Java"," James Gosling");
    }
}
```

```java
package com.ravi.factory_method;

public class BookDemo
{
    public static void main(String[] args)
    {
        Book book = Book.getBookObject();
        System.out.println(book);
    }
}
```

//Program on static factory method

```java
package com.ravi.factory_method;

import java.util.Scanner;

public class Customer
{
    private int id;
    private String name;
    private double bill;

    public Customer(int id, String name, double bill)
    {
        super();
        this.id = id;
        this.name = name;
        this.bill = bill;
    }

    @Override
    public String toString()
    {
        return "Customer [id=" + id + ", name=" + name + ", bill=" + bill + "]";
    }

    public static Customer getCustomerObject() //static factory method
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Customer Id :");
        int cid = Integer.parseInt(sc.nextLine());

        System.out.print("Enter Customer Name :");
        String cname = sc.nextLine();

        System.out.print("Enter Customer Bill :");
        double bill = Double.parseDouble(sc.nextLine());

        Customer c1 = new Customer(cid, cname, bill);

        return c1;
    }

}
```

```java
package com.ravi.factory_method;

import java.util.Scanner;

public class CustomerDemo {

    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("How many Customer Object :");
        int noOfObj = sc.nextInt();

        for(int i=1; i<=noOfObj; i++)
        {
           Customer cust =  Customer.getCustomerObject();
           System.out.println(cust);
        }
         sc.close();
    }

}
```

## Another Program on Static Factory Method :

```java
package com.ravi.factory_method;

import java.util.Scanner;

public class Product {
    private int id;
    private String name;
    private double price;

    public Product(int id, String name, double price)
    {
        super();
        this.id = id;
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString()
    {
        return "Product [id=" + id + ", name=" + name + ", price=" + price + "]";
    }

    public static Product getProductObject()
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Product Id :");
        int pid = Integer.parseInt(sc.nextLine());

        System.out.print("Enter Product Name :");
          String pname = sc.nextLine();

        System.out.print("Enter Product Price :");
        double price = sc.nextDouble();

        return new Product(pid, pname, price) ;

    }
}
```

```java
package com.ravi.factory_method;

import java.util.Scanner;

public class ProductDemo
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("How many Product Object :");
        int noOfObj = sc.nextInt();

        for(int i=1; i<=noOfObj; i++)
        {
          Product p1 = Product.getProductObject();
          System.out.println(p1);
        }
        sc.close();
    }

}
```