** Internally structure of System.gc();

```
class System
{
public static void gc()
 {
 Runtime.getRuntime().gc();
 }
}
```

Internally structure of Runtime class

```
public static void gc()
 {
 Runtime.getRuntime().gc();
 }
```

J.V.M खुदसे decision ले रहा है, destroy कर रहा है Objects को बिना request किये eg:

(4)
```
class A
{
int x,y;
static int c=0;
public void finalize()
 {
 System.out.println(++c);
 }
}
class Demo
{
public static void main(String ar[])
 {
 A a[]=new A[100000000];
 int i;
for(i=0;i<a.length;i++)
 {
 a[i]=new A(); → यहाँ memory create हुई
 a[i]=null; → और यह unreferenced
}}
```

O/P - 0
⋮

java.lang.OutOfMemory
Error: java heap space

NOTE: इस example में finalize method जितनी बार call hogi उतनी bar c की value increase hogi.

- इसमे कितने भी object destroy ho sakte hai koi problem nahi hai
- error, Exception] ये दोनो ही चीजे runtime पर ही आती है!
- Compile time पर Syntax error आती है!

*Difference between error and exception ?

Key	Error	Exception
Type	Classified as an unchecked type.	Classified as checked and unchecked.
Package	it belongs to java.lang.error	it belongs to java.lang.Exception.
Recoverable	It is iHRecoveRable (non recoveRable)	it is RecoveRable
Example	Out of Memory Error IO Error	Null Pointer Exception Sql Exception.
Types	Two types compile Time Exception	one Type Runtime Exception

ERROR	Exception
it is caused due to System crash or due to lack of resources.	it is caused due to some issues |mistakes in program code

it is unrecoverable	it is recoverable
there are no means to handle error by program code.	Exception can be handled using try, catch, throw throws, finally keyword.
if error occurs, then program get terminated	if exception occurs then it is thrown & some statement are executed from the catch block
Error are classified as unchecked type	Exception are classified as checked or unchecked.
Managed by class java.lang.Error	Managed by class java.lang.Exception.
eg stack overflow Out of Memory	eg: checked Exception No such Method class Not found.

(5)

```
class A
{
 int x, y;
void get(int a, int b)
 {
 x = a;
 y = b;
 }
void show()
 {
System.out.println("x="+x);
System.out.println("y="+y);
 }
}
```

```
class Demo
{
public static void main (string ar[])
 {
  A a1=new A();
  a1.get (10,20);
  A a2=a1;
  a1.show();
  a2.show();
  a2.x=55;
  a1.show();
  a2.show();
 }
}
```

```
o/p:- x=10
      y=20
      x=10
      y=20
      x=55
      y=20
      x=55
      y=20
```

CLONING

cloning :- the object cloning is a way to create exact copy of an object. the clone() method of Object class is used to clone an object.

The java.lang.Cloneable interface must be implemented by the class whose object clone we want to create. if we don't implement cloneable interface, clone() method generates CloneNotSupport Exception.

the clone() method is dfined in the Object class. syntax of the clone() method is as follows :-

```
Protected object clone() throws ClassNotSupported
                                Exception
```

NOTE :- i) clone() method object class के पास है!

ii) object class की clone method का return type object होता है।

iii) Object class ki clone method protected hai.

iv) हमें जिस class के object का clone banana hai तो वो class clonable interface से implement hona compulsory hai। यदि हम ऐसा नहीं करते हैं तो cloneNotSupported message aakar display hoga।

v) ClassNotSupported checked exception hai?

⑥
```
class A
{
int x,y;
void get(int a,int b)
{
x=a;
y=b;
}
void show()
{
System.out.println("x="+x);
System.out.println("y="+y);
}
}
class Demo
{
public static void main(String args[])
{
A a1=new A();
a1.get(10,20);
a1.show();
A a2=a1.clone();
a2.show();
a2.x=55;
a1.show();
a2.show();
}
}
```

O/p- error: clone() has protected access in Object
A a2=a1.clone();
error: incompatible types: Object cannot be convert to A
A a2=a1.clone();

NOTE : clone() method protected hai or protected data yaa to ~~isukee~~ class me access hota hai yaa to uskee child class me

A a2 = a1.clone();

↙ me subclass hai object ke liye

clone() method object class ke paas hai aur object sabse superclass hai to bhi A uske liye subclass hai usme store nahi hoga.

* cloning example : *

```
(1) class A implements Cloneable
    {
     int x,y;
    void get(int a, int b)
     {
     x=a;
     y=b;
    }
    void show()
    {
    System.out.println("x="+x);
    System.out.println("y="+y);
    }
    public Object clone() throws CloneNotSupportedException
     {
     return super.clone(); → yahaa super.clone() return karaana padega tab object class kee clone() call hogi
     }
    }
    class demo
    {
    public static void main(String arg[]) throws CloneNotSupported
                                                  Exception
     {
      A a1=new A();
```

* protected native java.lang.Objet clone() throws java.lang.CloneNotSupportedException

* Example jo exception compile par aa rahee hai use throws keyword kee help se remove karne ke baad bhee run time par aayegee kyonki uskee requirement fullfill nahee kee hai !

```
a1.get(10,20);
A a2=(A)a1.clone();
a1.show();
a2.show();
a2.x=55;
a1.show();
a2.show();
}
}
```

o/p - x=10
y=20
x=10
y=20
x=10
y=20
x=55
y=20

यहाँ clone() object class की है और class A subclass है object की तो explicity type casting करनी पड़ेगी

NOTE:- Singleton class में constructor को private इसलिए बनाया है क्योंकि उसे call न कर सकते अगर dyaut constructor होता तो हर object के corresponding बार-बार call होता !

(8)
```
class Demo implements Cloneable
{
int x,y;
void get(int a, int b)
{
x=a;
y=b;
}
void show()
{
System.out.println("x="+x);
System.out.println("y="+y);
}
public static void main(String arg[]) throws CloneNotSupportedException
{
Demo a1=new Demo();
a1.get(10,20);
Demo a2=(Demo)a1.clone();
a1.show();
a2.show();
a2.x=55;
a1.show();
a2.show();
}}
```

o/p - x=10
y=20
x=10
y=20
x=10
y=20
x=55
y=20

(9)

```
class B
{
int i=10;
}
class A
{
int j=20;
int k=30;
B b=new B();
}
class Demo
{
public static void main(String ar[])
{
A a1=new A();
System.out.println(a1.j);
System.out.println(a1.k);
System.out.println(a1.b.i);
}
}
```

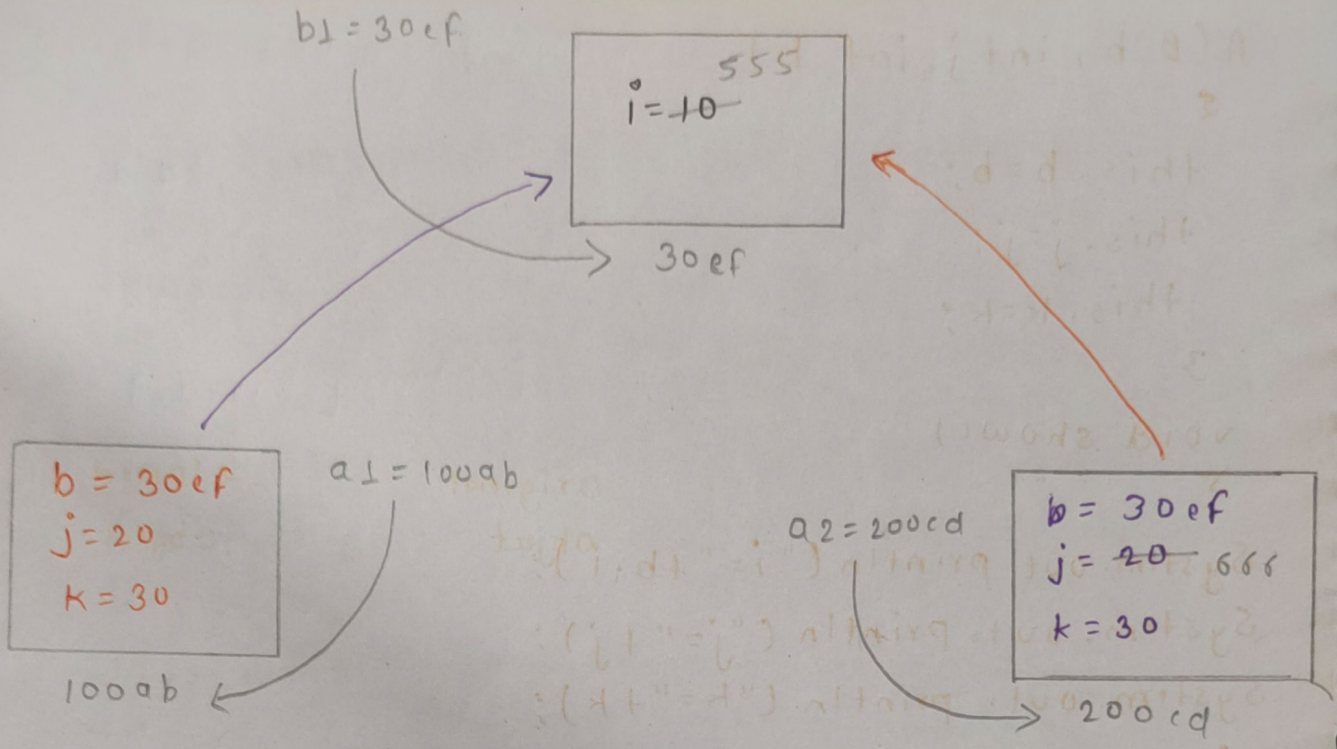
O/P - 20
30
10

(10)

```
class B
{
int i;
B(int i)
{
this.i=i;
}
}
class A implements Cloneable
{
B b;
int j;
int* k;
```

```
A(B b, int j, int k)
{
 this.b=b;
 this.j=j;
 this.k=k;
}
void show()
{
System.out.println("i="+b.i);
System.out.println("j="+j);
System.out.println("k="+k);
}
public Object.Clone() throws CloneNotSupportedException
{
return super.clone();
}
}
class Demo
{
public static void main(String arg[]) throws CloneNotSupported
                                              Exception
{
B b1=new B(10);
A a1= new A(b1,20,30);
A a2=(A) a1.Clone();
a1.show();
System.out.println("..........");
a2.show();
System.out.println("..........");
a2.b.i=555;
a2.j=666;
a1.show();
System.out.println("..........");
a2.show();
}
}
```

O/p- i=10
j=20
k=30
.....
i=10
j=20
k=30
.....
i=555
j=20
k=30
.....
i=555
j=666
k=30

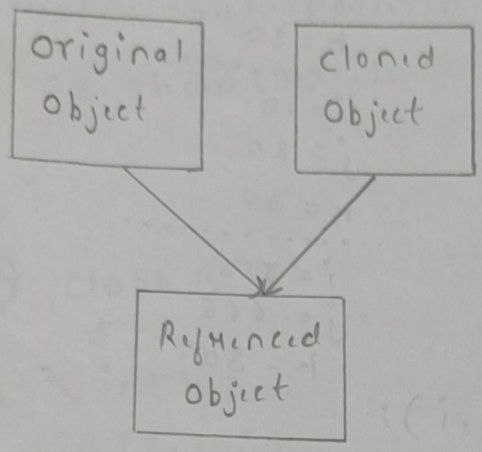

b1 = 30ef
i = 10 555
30ef
b = 30ef
j = 20
k = 30
a1 = 100ab
100ab
a2 = 200cd
b = 30ef
j = 20 666
k = 30
200cd

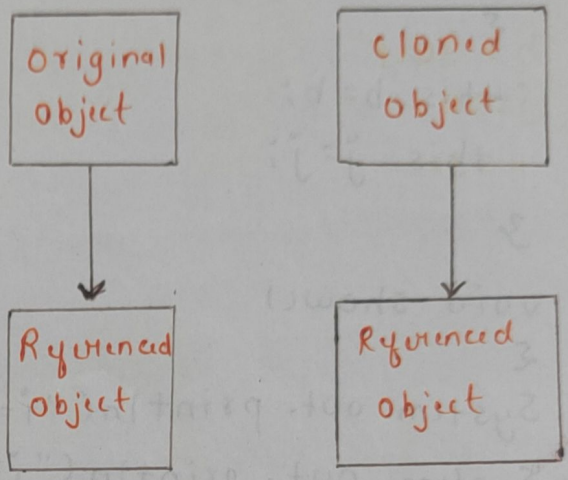
01/08/23
Friday

Types of Cloning :- there are two types of cloning

i) shallow cloning.

ii) dup cloning.





→ दोनो एक ही memory को point kar rahe hai isliye इसे shallow cloning कहते हैं

→ Object class की clone method by dyault shallow cloning को hi support करती है। क्योंकि तो भी by dyault 2 object के लिए common ref. object को point karti hai

User dyine cloning को ही dup cloning कहा जाता है!

⇒ the process of creating exactly the independent duplicate objects in the heap memory & manually assigning the values of the second object where values are supposed to be copied is called deep cloning.

* Deep cloning Example *

```
(1) class B
    {
    int i;
    B (int i)
    {
    this.i=i;
    }
}
```

```
class A implements Cloneable
 {
  B b;
  int j;
  int k;
A(B b, int j, int k)
 {
  this.b=b;
  this.j=j;
 }
void show()
{
System.out.println("i="+b.i);
System.out.println("j="+j);
System.out.println("k="+k);
}
public Object clone() throws CloneNotSupportedException
 {
  B b5=new B(b.i);
   A a5=new A(b5,j,k);
   return a5;
 }
}
Class Demo
 {
public static void main(string ar[]) throws CloneNot
                                    SupportedExcaption
 {
  B b1=new B(10);
  A a1=new A(b1,20,30);
   A a2=(A)a1.Clone();

   a1.show();
  System.out.println("..........");
```

o/p- i=10
j=20
k=30
.....
i=10
j=20
k=30
.....
i=10
j=20
k=30
.....
i=555
j=666
k=30

```
a2.show();
System.out.println("..........");
a2.b.i=555;
a2.j=666;
a1.show();
System.out.println("..........");
a2.show();
}
}
```

(2)

```
class B
{
int i;
B(int i)
{
this.i=i;
}
}
class A
{
B b;
int j,k;
A(B b,int j,int k)
{
this.b=b;
this.j=j;
this.k=k;
}
void show()
{
System.out.println("i="+b.i);
System.out.println("j="+j);
System.out.println("k="+k);
```

```
public Object clone()
  {
   B b5=new B(b.i);
    A a5=new A(b5,j,k);
    return a5;
  }
}

class Demo
{
public static void main(String ar[])
  {
   B b1=new B(10);
   A a1=new A(b1,20,30);
   A a2=(A)a1.clone();
  a1.show();
System.out.println("........");
  a2.show();
System.out.println("........");
  a2.b.i=555;
  a2.j=666;
  a1.show();
System.out.println("........");
  a2.show();
  }
}
```

O/P

```
i=10
j=20
k=30
.........
i=10
j=20
k=30
.........
i=10
j=20
k=30
.........
i=555
j=666
k=30
```

NOTE Class.forName → static method

↙ Predefined class

* किसी भी class में कितनी methods हैं उन्हें show कराने का program ?

(3)
```
import java.lang.reflect.*;
class Demo
{
public static void main(String ar[])
{
Object o=new Object();
Class c=o.getClass();
Method m[]=c.getDeclaredMethods();
for(Method m1:m)
{
System.out.println(m1);
}
}
}
```

o/p- All object class method show

(4) String class all methods show program

```
import java.lang.reflect.*;
class Demo
{
public static void main(String ar[])
{
String o=new String();
Class c=o.getClass();
Method m[]=c.getDeclaredMethods();
for(Method m1:m)
{
System.out.println(m1);
}
}
}
```

o/p- all string class method show

* किसी भी class में कितनी methods है उसे count करने का code

⑤
```
import java.lang.reflect.*;
class Demo
{
public static void main(String ar[])
{
 Object o=new Object();
 Class c=o.getClass();
Method m[]=c.getDeclaredMethods();
 int count=0;
for(Method M1:m)
{
 count++;
}
System.out.println("total methods="+count);
}
}
```

O/p - 12

String Class methods count program ?

⑥
```
import java.lang.reflect.*;
class Demo
{
public static void main(String ar[])
{
 String o=new String();
 Class c=o.getClass();
Method m[]=c.getDeclaredMethods();
 int count=0;
for(Method m1:m)
{
 count++;
}
```

```
System.out.println(count);
}
}
```

o/p - 77

= getDeclaredMethods :- The getDeclaredMethods() method of java.lang.Class, class is used to get the Method of this class, which are the method that are private public, protected or default and its members or the members of its member classes and interfaces but not the not inherited methods.

throws :- throws keyword कभी Exception को handle नहीं करता hai exception को तो सिर्फ current method से calling method के pass forward karta hai. Exception को handle सिर्फ try, catch or finally keyword की help से कर सकते हैं!

(7)
```
import java.lang.reflect.*;
class Demo
{
public static void main(String ar[]) throws ClassNotFoundException
{
Class c = Class.forName("java.lang.Object");
Method m[] = c.getDeclaredMethods();
int count = 0;
for(Method m1 : m)
{
System.out.println(m1);
count++;
}
System.out.println(count);
}
}
```

o/p - 12

ये यहाँ पर इसने String में लिखना है तो यहाँ पर exception आने के chances है इसलिए compile time पर warning करेगा और इसने throws keyword की help से exception को देना पड़ेगा

NOTE :- Double ("") में लिखा हुआ runtime पर check hoga

. हम दो type से class के अंदर की method show kara सकते हैं

(i) Object o=new Object();
Class c= o.getClass();

(ii) Class c = Class.forName("java.lang.Object");

NOTE. 2nd वाले case में exception आने के chance है, USER ने अगर ("java.lang.object") यह string में कुछ गलत pass कर दिया तो

2nd type का benefit भी है हम इसमें Class.forName(arg[0]) argument pass कराके बाद में भी input ले सकते है।

What is Boxing?

Boxing :- Boxing is the process of converting a value type to the object or to any interface type implemented by this value type. when the common langu runtime boxes of value type it wraps the value inside. or System.

```
class Boxing
{
public static void main (String arg[])
 {
  int i=10;                          O/P- go
  Object o=i;                             10
    i= go
System.out.println(i);
System.out.println(o);
}}
```

What is unboxing?

Unboxing :- Unboxing is a explicity conversion from the type object to a value type from an interface type to a value type that implements the interface. An unboxing operation consists of checking operation consists of checking the object instance to make sure that is boxed value of the give value

```
class unboxing
{
public static void main (string arg[])
{
int i=10;                                   o/p - 10
Object o=i;  Integer x=new Integer(10)          10
i=(int)o;      int y=x.intValue();
System.out.println(o);
System.out.println(i);
}
}
```

Autoboxing :- Primitive type ke corresponding wrapper class me automatic conversion kehlata hai. Ye concept version 1.5 se aaya hai isse pehle work nhi karta tha!

```
Class Demo
{
public static void main (String arg[])
{                                              o/p - 10
int x=10;                                          10
Integer y=x;
System.out.println(x);
System.out.println(y);
}
}
```

Autounboxing :- the automatic conversion of wrapper class type into corresponding primitive type. is known as autounboxing.

```
Class Demo
{
public static void main (String arg[])      o/p - 10
{                                                10
Integer x=new Integer(10);
int y=x;
```

```
System.out.println(x);
System.out.println(y);
}
}
```

* What is method overriding in java?

In any object-oriented programming language overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-class or parent classes. when a method in a subclass has the same name, same parameters or signature and same return type (or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

NOTE :- Method overriding is one of the way by which java achieve Run time polymorphism

the version of a Method that is executed will be determined by the object that is used to invoke the method, then the version in the parent class will be executed but if an object of the subclass is used to invoke the method then the version in the child class will be executed In other words it is the type of the object being referred to (not the type of Reference variable) that determines which version of an overriden Method will be executed.

Q. What are Rules of method overriding in java?

* Rules of Method overriding in Java *

- In java a method can only be written (overridden) in subclass not in same class.
- the argument list should be exactly same as that of overridden method.
- the access level cannot be more restrictive then the overridden method access level.

eg: If the superclass method is declared public then the overriding method is the sub class cannot be either private, default or protected.

private → default → protected → public

```
class A
{
int show()
 {
System.out.println("ram");
  return 100;
 }
}
class B extends A
 {
  long show()
  {
  System.out.println("sita");
   return 10;
  }
}
class Demo
{
public static void main(String args[])
 {
    A a1=new A();
}   a1.show();
```

Error: show() in B cannot override show() in A

NOTE: return type या तो same होना चाहिए या तो superclass में parent और subclass में child relation होना चाहिए।

* The return type should be same or a subtype of the Return type declared in the original overriden method in the super class.

```
class A
{
Object show()
{
System.out.println("ram");
return "class A";
}
}
class B extends A
{
System.out.println("sita");
return "class B";
}
}
class Demo
{
public static void main(string args[])
{
A a1=new B();
System.out.println(a1.show());
}}
```

O/P- class B
sita

NOTE: ये facility version 1.5 से आई है। इस facility का नाम covariant method overriding है

```
class A
{
String show()
{
System.out.println("
```

```
    return "class A";
    }
}
class B extends A
    {
    Object show()
    {
    System.out.println("sita");
    return "class B";
    }
}
class Demo
{
public static void main(String ar[])
{
A a1=new B();
System.out.println(a1.show());
}}
```

Error: show() in B cannot override show()

return type Object is not compatible with String

```
class A
{
int show()
{
System.out.print("ram");
    return 10;
}
}
class B extends A
    {
    byte show()
    {
System.out.println("sita");
    return 100;
}
```

```
class Demo
{
public static void main(String arg[])
{
  A a1=new B();
System.out.print(a1.show());
}
}
```

O/p- error: show() in B cannot
override show() in A byte

return type byte is not compatible
with int

02/09/23
Saturday

- Instance Method can be overridden only if they are inherited by the subclass.
- A method declared final cannot be overriden
- A Method declared static cannot be overriden but can be re-declared.
- A subclass within the same package can override any superclass method that is not declared private final or static
- A subclass in a different package can only override the non-final method declared public or protected.

यदि superclass की method किसी exception को throws नहीं कर रही है तो subclass में हम नहीं कर सकते हैं किसी भी exception को throws नहीं कर सकते हैं।

(1)
```
class A
{
void show()
{
System.out.println("ram");
}
}

class B extends A
{
void show() throws Exception
{
System.out.println("sita");
}
}

class Demo
{
public static void main(String args[])
{
B b=new B();
b.show();
}
}
```

O/P- error show() in B cannot override show in A void show() throws Exception

(2)

```
class A
{
void show() throws Exception
 {
 System.out.println("ram");
 }
}
class B extends A
 {
 void show() throws IOException
 {
 System.out.println("sita");
 }
}
 class Demo
 {
 public static void main(String arg[])
  {
   B b=new B();
   b.show();
  }
}
```

NOTE: Super class की method जिस type की exception को through कर रही है उसी type की subclass भी करना चाहिए या child type की होना चाहिए।

o/p - sita

(3)

```
class A
 {
 void show() throws IOException
 {
System.out.println("ram");
 }
}
class B extends A
 {
 void show() throws Exception
 {
 System.out.println("sita");
 }
}
 class Demo
 {
```

o/p - error : show in B cannot override show A

```
public static void main(String ar[])
  {
   B b=new B();
    b.show();
  }
}
```

* ये rule applicable है केवल checked exception के लिए
* unchecked exception के लिए कोई भी rule नहीं है कहीं भी कैसे भी exception को throw कर सकते है।

(4)
```
class A
{
void show()
{
System.out.println("ram");
}
}
class B extends A
{
void show() throws ArithmeticException
{
System.out.println("sita");
}
}
class Demo
{
public static void main(String ar[])
{
  B b=new B();
   b.show();
}
}
```

o/p- sita

- there are many rules if we talk about method overriding with exception handling. the rules are as follows.
- if the superclass method does not declare an exception subclass overriden method cannot the checked exception but it can declare unchecked exception
- if the superclass method declares an exception subclass overriden method can declare same subclass exception

or no exception but cannot declare parent exception.

Q.2 Can we overRide a method which throws unchecked exception?

Yes

there is no RestRiction on unchecked exception while overRiding on the other hand in the case of checked exception an overRiding exception cannot throw a checked exception which comes higher in type hierarchy

eg- if original method is throwing IOException then overRiding method cannot throw java.lang.Exception or java.lang.throwable

```
class A
{
static void show()
{
System.out.println("ram");
}}
class B extends A
{
void show()
{
System.out.println("sita");
}}
class Demo
{
public static void main(String ar[])
{
A a=new B();
a.show();
}
}
```

o/p - error: show() in B cannot overRide show in A

```
class A
{
 static void show()
 {
 System.out.println("ram");
 }
}

class B extends A
{
 static void show()
 {
 System.out.println("sita");
 }
}

class Demo
{
public static void main(String ar[])
 {
 A a=new B();
 a.show();
 }
}
```

o/p. sita

Method Hiding -:- work at reference variable corresponding

Q3. Can we override a non-static method as static in java?

Hiding happens when both methods in the pair are static methods, overriding happens when both methods in the pair are instance methods. when one of the two is a static method and the other one is an instance method. Java considers it an error. it does not matter if the instance method is final or not, it also does not matter if the static method is in the base or in the derived class. Java calls it an error either way.

Q 4. Can we override static method in java?

No you cannot override static method in java because method overriding is based upon dynamic binding at runtime and static methods are bonded using static binding at compile time.

you can declare a method with same name and method signature in subclass which does look like you can override static method in java but in reality, that is method hiding

if you use reference variable of type subclass the new (subclass) method will be called, but if you use reference variable of superclass than old (super class) method will be called. in short you cannot override static method in java.

* Private, final और static method को override नहीं कर सकते है

Q. Method Overloading in Java?

Method Overloading :- Whenever same method name is existing multiple time in the same class with different number of parameter or different order of parameters or different type of parameters is knows as method overloading.

For eg :- System.out.println();
is overloaded as we have 9 or 10 println() method each accepting a different type of parameter.

Q. 2) What are rules of method overloading in java?

Rules of Method overloading :- One of the rules of method overloading in java is that method signature should be different i.e either the

number of arguments or the type of argument.
Simply changing the return type of two method will not result in overloading instead the compiler will throw an error.

```
class A
{
void show()
{
System.out.println("class A");
}
int show()
{
return 100;
}
}

class Demo
{
public static void main(String args[])
{
A a=new A();
System.out.println(a.show());
}
}
```

Q.3) can we overloaded a static method in java?
Yes we can overload a static method in java. you can declare as many static method of the same name as you wish provided all of them have different method signature.

Static, Private, Final } override ✗ overload ✓

Q.4) can be overload the main method in java?
Yes we can overload main method in java. But the program doesn't execute the overloaded

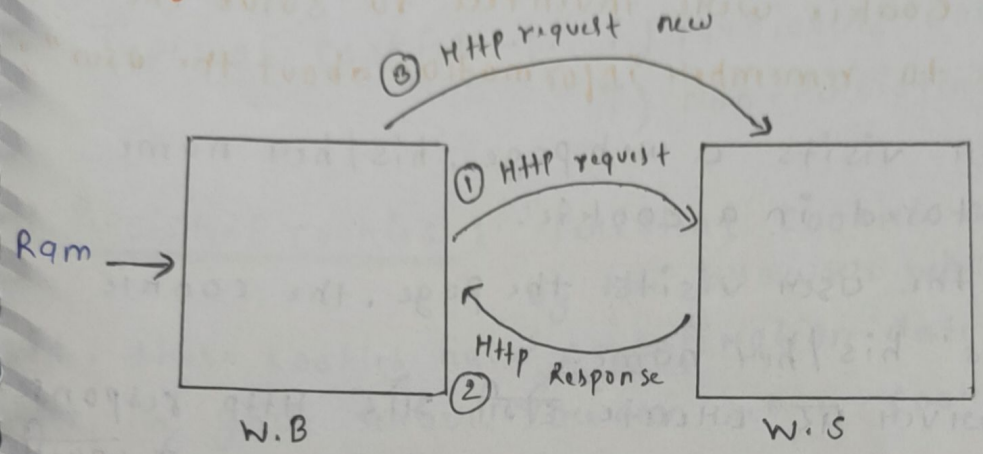
main method when we run our program we have to call the overloaded main method from the actual main method that means main method acts as an entry point for the java interpreter to start the execute of the application. where as a loaded main need to be called from main()

```
class Demo
{
public static void main()
 {
 System.out.println("ram");
}
public static void main (String arg[])
 {
 main();
System.out.println("main method");
}
}
```

o/p - ram
main method.

04/09/23
Monday –

★ Http is a stateless protocal (forgetfull) वो भूल जाता है कि उसने क्या किया था जैसे Http ने एक बार ram के corresponding request भेजी तो new req. मानेगा लेकिन again ram के corresponding भेजी तब भी new request ही मानेगा !



यहाँ 3rd step में ram फिर से Http request भेजेगा तो Http request को server new request ही मानेगा और new की तरह ही treat करेगा उसे यह नहीं पता है कि पहले भी request ram ने ही भेजी थी !

Types of user :- there are two type of user :-

i.) जो login हो चुका है।
ii) जो login नहीं हुआ है।

"particular user को recognize करने के purpose से session tracking techniques का use किया जाता है। session tracking technique को ही session Management technique भी कहा जाता है

There are four types of techniques :-

(i) Cookies
(ii) Session
(iii) Hidden form Field.
(iv) URL Rewriting.

Cookies :- Cookies are data stored in small text files, on our computer.

When a web server has sent a webpage to a browser, the connection is shut down, and the server forgets everything about the user.

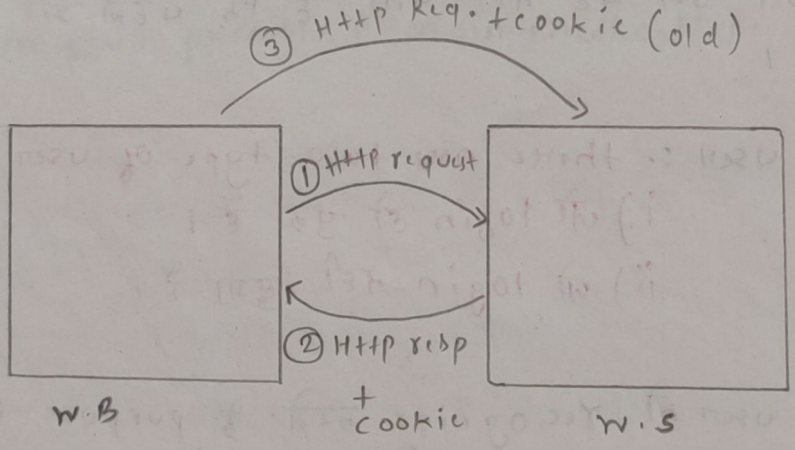
Cookie were invented to solve the problem "how to remember information about the user";

- When a user visits a web page, his/her name can be stored in a cookie.
- Next time the user visits the page, the cookie "remembers" his/her name.

i) Cookie Server पर create होगी और Http response के साथ Http response + cookie web Browser मे जाएगी

ii) Cookie create Server पर होगी

iii) cookie store Browser पर होगी



जब Http Response + cookie browser के पास आएगी और double (दोबारा) से वही user Request भेजेगी तो उसके साथ cookie भी Server पर जाएगी तो और Server समझ जाएगा की जो request कर रहा है वो old user है।

इसके बाद जितनी भी Http Req. या Http response भेजेंगे cookie भी साथ मे travel करने वाली है।

Cookie create कब होगी ?

if (username & password)

{

}

जब यदि username और password सही है तभी cookie create होना चाहिए हम if में ऐसी condition डालेंगे।
यदि cookie create हो चुकी है तब ही menu वाला page open होना चाहिए otherwise नहीं होना चाहिए।

Types of cookies -: i) Persistent cookies.
ii) Non-Persistent cookies.

i Persistent cookies :- "Persistent cookies are not deleted by the browser when the user close it, these cookies have an expiration date that you can set in your server. You can set a cookie to expire in a day or ten year.

ऐसी cookie जो multiple session के लिए valid होती है उसे persistent cookie कहते हैं like (gmail) हम जब तक इसे खुदसे logout नहीं करेंगे browser हमे open करने की permission देता रहेगा हम कितनी भी बार इसे open कर सकते हैं

हम Persistent cookies की expiry भी set कर सकते हैं कि वो हमें कितने time तक रखना है। उसके लिए method है :-

Cookie ck = new Cookie ("", "");

महाँ cookie का object बनेगा — variable name — variable value

expiry set Method

ck.setMaxAge(60x30);

30 minute expiry set

cookie 30 minute के बाद automatically expire हो जायेगी

यदि हमें 30 din की expiry set करना है तो :-

ck.setMaxAge(60*60x24x30)

जब आपसे पास cookie बनकर तैयार हो चुकी है तो cookie को web server से web browser के पास भेजना है तो:

response.addCookie(ck);

ऐसा करने से cookie browser के पास चली जाएगी।

Non-Persistent cookies :- "Non-Persistent cookies are otherwise called as tempory cookies. they are active as long as the browser remails active. they are also called as session based cookies. Once the browser is closed the cookies vanishes automatically.

ऐसी cookies जो सिर्फ single session (one time) के लिए ही valid होती है like (banking से related website). एक बार Browser close करने पर automatically logout हो जाता है।

Cookie create कि login.java project code में

```
if(rs.next())
{
Cookie ck = new Cookie("un","ram");
ck.setMaxAge(60*30);
response.addCookie(ck);
response.sendRedirect("Menu");
```

30 minute के बाद automatically expire हो जाएगा logout.

Cookie	Session
i Cookie are client-side files on a local Computer that hold user information	i) Session are server side files that contain user data

iii) Cookies end on the lifetime set by the user	When the user quits the browser or logs out of the programmed, the session is over.
iii) it can only store a certain amount of info	it can hold an indefinite quantity of data.
iv) the browser's cookies have a maximum capacity of 4KB	we can keep as much data as we like within a session, however there is a maximum memory restriction of 128 MB that a script may consume at one time.
v) Because cookies are kept on the local computer we don't need to run a function to start them.	To begin the session we must use the session start() method.
vi) cookies are not secured.	Session are more secured compare than cookies
vii) cookies stored data in text file	session save data in encrypted form.
viii) cookies stored on a limited data	Session stored a unlimited data.

Q यदि हमने cookie की expiry set नहीं की है तो cookie कितने time तक रहेगी ?

Browser session and cookie will destroy

Browser close करते ही cookie destroy हो जाएगी

यदि हमने उसकी expiry set नहीं की है तो ।

हमें हमारे project के code में यदि username और password सही है तो cookie create होनी चाहिए का code करना है तो :-

Login.java

```
if(rs.next())
{
Cookie ck= new Cookie("un","ram");
ck.setMaxAge(60*30);
response.sendRedirect("mymenu");
addCookie(ck);
}
else
{
out.println("invalid username and password");
}
```

* इसके बाद हमे menu.html यहाँ work नही करेगा इसीलिए हमें menu.java servlet बनाकर तैयार करना पड़ेगा क्योंकि cookie वाला code java का है.

Menu.java

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Menu extends HttpServlet
{
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
```

```
{
Cookie ck[]= Request.getCookies();
PrintWriter out= response.getWriter();
if(ck!= null)
{

}
else
{
out.println("enter corred data");
}
out.close();
}
}
```

cookies को fetch करने के लिए get cookies() है। इसे array type के Object में store करायेंगे

ck में value है तो if चलेगा।

NOTE:- Cookie ck[] = request.getCookies(); cookies के data को fetch करने के लिए ये method provide की है

```
if(ck!=null)
{
-||-
}
else
{
}
```

→ यहाँ यदि सही username और password enter किया है तब ही cookie create होगी और जब cookie create होगी तब ही ये condition true होगी और इसके अंदर का code execute hoga

अब यदि हम चाहते हैं कि Login page पर जो logout button है उस पर click करने पर logout भी हो जाए और जो cookie create हुई है वो भी delete हो जाए तो हमें एक और logout.java Servlet page वगैरह तैयार करना होगा!

```
{
Cookie ck[]= request.getCookies();
PrintWriter out= response.getWriter();
if(ck!= null)
  {


  }
else
{
out.println("enter correct data");
}
out.close();
}
}
```

→ cookies को fetch करने के लिए get cookies() है। इसे array type के object में store कराएंगे

ck में value है तो if चलेगा

NOTE:- Cookie ck[] = request.getCookies(); cookies के data को fetch करने के लिए ये method provide की है

```
if(ck!= null)
  {
-||-
  }
else
  {
  }
```

→ यहाँ यदि सही username और password enter किया है तब ही cookie create होगी और जब cookie create होगी तब ही ये condition true होगी और इसके अंदर का code execute होगा

अब यदि हम चाहते हैं कि login page पर जो logout button है उस पर click करने पर logout भी हो जाए और जो cookie create हुई है वो भी delete हो पाए तो हमें एक और logout.java servlet वगैरह तैयार करना होगा!

LogOut

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class LogOut extends HttpServlet
{
public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException
{
Cookie ck=new Cookie("un","");
ck.setMaxAge(0);
response.addCookie(ck);
response.sendRedirect("Login.html");
}
}
```

STEP for COOKIES

i) Cookie का object बनायेंगे।

```
Cookie ck=new Cookie();
```

ii) Cookie की expiry set करेंगे

```
ck.setMaxAge();
```

iii) response object को browser के पास भेजेंगे।

```
response.addCookie(ck);
```

iv) Cookie को access करेंगे।

```
request.getCookies() से
Cookie ck[] = request.getCookies();
```

v) Cookie को destroy करेंगे।

```
ck.setMaxAge(0);
```

LogOut

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class LogOut extends HttpServlet
{
public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException
{
Cookie ck=new Cookie("un","");
ck.setMaxAge(0);
response.addCookie(ck);
response.sendRedirect("Login.html");
}
}
```

STEP for COOKIES

i) Cookie का object बनायेंगे।

```
Cookie ck=new Cookie();
```

ii) Cookie की expiry set करेंगे

```
ck.setMaxAge();
```

iii) Response object को browser के पास भेजेंगे।

```
response.addCookie(ck);
```

iv) Cookie को access करेंगे।

```
request.getCookies() से
Cookie ck[] = request.getCookies();
```

v) Cookie को destroy करेंगे।

```
ck.setMaxAge(0);
```

Cookie	Session
1. Cookie में limited data ही store हो पाता है	Session में unlimited data store हो जाता है
2. Cookie में specific data ही store करते हैं (text file)	Session में data को encrypted form में save कर सकते हैं (images, animations etc).
3. Cookie की expiry set कर सकते हैं	Session की expiry set नहीं कर सकते हैं
4. Cookie Client Side पर होती है और store browser में	Session Server में store होता है
5. Cookie is not secured	Session is secured.
6. Cookie में हम expiry set करके उसे end कर सकते हैं	Session browser session end होते ही end हो जाता है

Interview

ऐसी कौन सी condition है जहाँ हम session का use करेंगे और ऐसी कौन सी condition है जहाँ हम cookie का use करेंगे?

getName() and getValue() method की help से हम variable name और variable value को web page पर display करा सकते हैं?

My memo

out.print("<h1>welcome</h1>"+ck[0].getName());

out.print("<h1>welcome</h1>"+ck[0].getValue());

कहीं भी doubt आगे भी कोई problem आए cookie में getName() method ही solve करेगी।

05/09/22

Tuesday

Session :- The time interval in which two System (i.e) the client and the server) communicate with each other can be termed as a session. in simple terms a session is a state consisting of several requests and response between the client and the server

why do we use Session ?

A session is a way to store information (in variable) to be used across multiple pages. unlike a cookie the information is not stored on the users computer.

यदि session को create करना है तो :-

request.getSession() method ka use करना पड़ेगा, इसका return type HttpSession है !

```
HttpSession session = request.getSession();
```

Session में data store करने के लिए method provide की है - setAttribute()

```
Session.setAttribute("uname", s1);
```

Login.java

```
if(rs.next())
{
HttpSession session = request.getSession();
Session.setAttribute("uname", s1);
Session..sendRedirect("Menu");
}
```

जितने भी pages पर session का code करना hai उनमे सबसे पहले session का object बनाकर तैयार करना पड़ेगा।

Menu.java

```
HttpSession session = request.getSession();
String s1 = (string) session.getAttribute("uname");
PrintWriter out = response.getWriter();
  if(s1 != null)
  {
  out.println("<h1> welcome "+s1+" </h1>");
  }
```

Data को session से fetch करने के लिए method है -
getAttribute();

```
Session.getAttribute("uname");
String s1 = session.getAttribute("uname"); → exception obj cannot be convert to string
```

getAttribute() method का return type Object hai or हम String में store करना रहे हैं Object super class है और string subclass इसलिए type cast करना पड़ेगा

```
String s1 = (string) session.getAttribute("un");
```

Logout.java

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    IOException, ServletExaption.
  {
  HttpSession session = request.getSession();
  session.invalidate();
```

response.sendRedirect("Login.html");

}

Session ko destroy करने के लिए method है!

Session.invalidate();

HttpSession session = request.getSession(true);

or

Http session session = request.getSession(false);

में क्या difference hai?

Ans- Http session session = request.getSession() यहाँ parameter में कुछ नहीं लिखा है इसका मतलब true है।

request.getSession(true) - ये method session का existence check करती है कि session exist कर रहा है या नहीं कर रहा है। यदि session already exist कर रहा है तो request.getsession() method उसी session के object को return कर देगा।

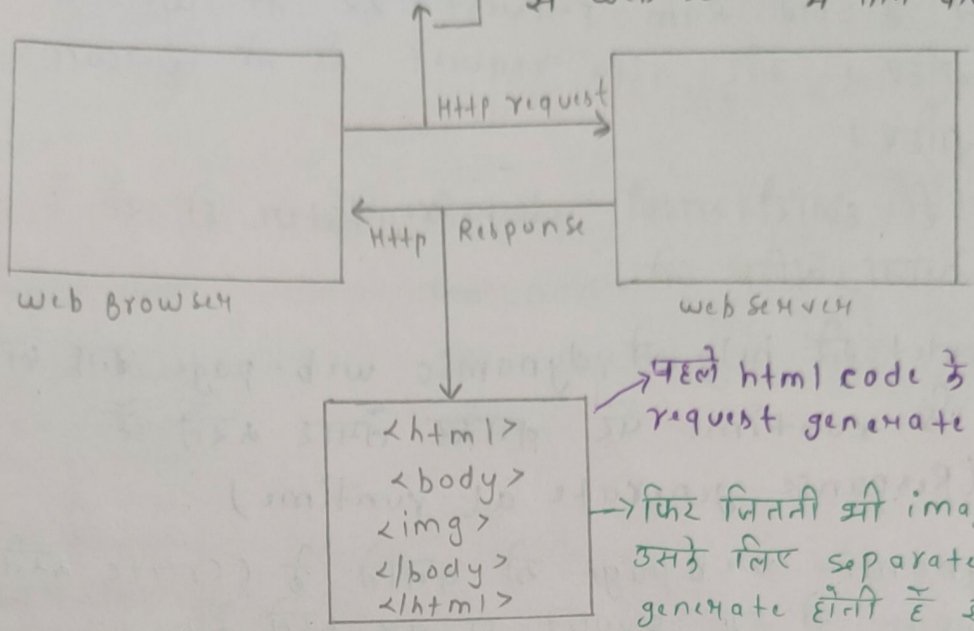
Otherwise request.getSession(true) method नये session के object को return करेगा।

request.getSession(false) :- अगर पहले से session का object exist कर रहा है तो ये उसी object को return करेगा otherwise ये null return करेगा!

06/09/23
Wednesday -

i) url - किस page के लिए data जा रहा है।
ii) Method-name. - form की method का name 1. get 2. post
iii) form data

Request करते समय web browser से web server में तीन चीजे जाती है,



पहले html code के लिए request generate hogi

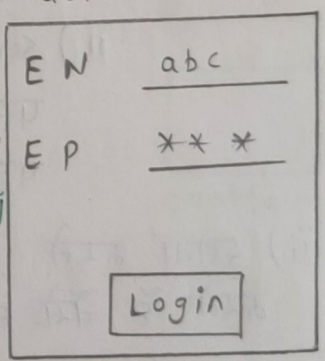
फिर जितनी भी images है उसके लिए separate request generate होती है इसीलिए कभी-कभी images load होने

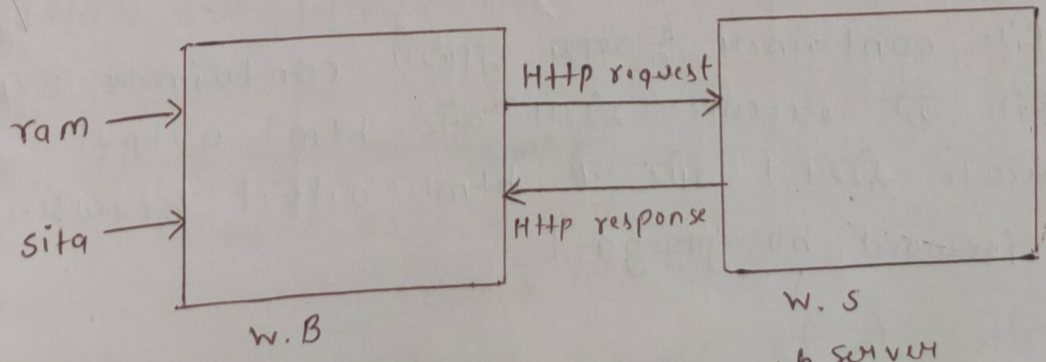
Get :- Get Method में पूरा का पूरा data url में भी time लगता है through होता हुआ जाता है। id, password सब show होते है। इसलिए id is not secure.

like - lg? un=abc &up=123

Get Method ka use हम कभी भी security के purpose से नही कर सकते है।

* Post :- post method में data page के url से होता हुआ नही जाता है। वो internal page के body part से होता हुआ server पर जाता है। it is secure जब भी हमे security के purpose से use करना है तो post method ka use करेंगे।





i) Ram ने request किया Demo.html page के लिए तो Server पर होगा तो आ जाएगा।
फिर sita ने request की तब भी same आएगा लेकिन हम चाहते हैं, जब Ram request करे तो welcome Ram आना चाहिए। और sita request करे तो welcome sita आना चाहिए।

ii) ऐसे ही हजारों, लाखों users register हो रहे हैं सबका नाम आना चाहिए तो;

iii) Servlet की help से dynamic web-page बना सकते हैं जो कि runtime पर बनकर तैयार करते हैं
(Response generate at runtime)

iv) Dynamic webpage को बनाने के (create करने के) purpose से हम Servlet का use करते हैं।

*** web page को runtime पर बनाने के purpose से Servlet का use करते हैं।

SERVLET

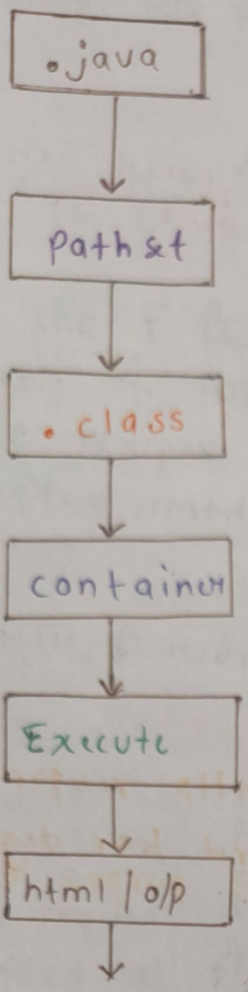
i) Servlet के program को बनाने के लिए हमे (.java) file को create करना hoga

ii) servlet के program को compile करने से पहले servlet-api का classpath set करना पड़ता है।

iii) इतना करने के बाद normal java program को जैसे compile करते हैं वैसे ही servlet program को compile करना hoga

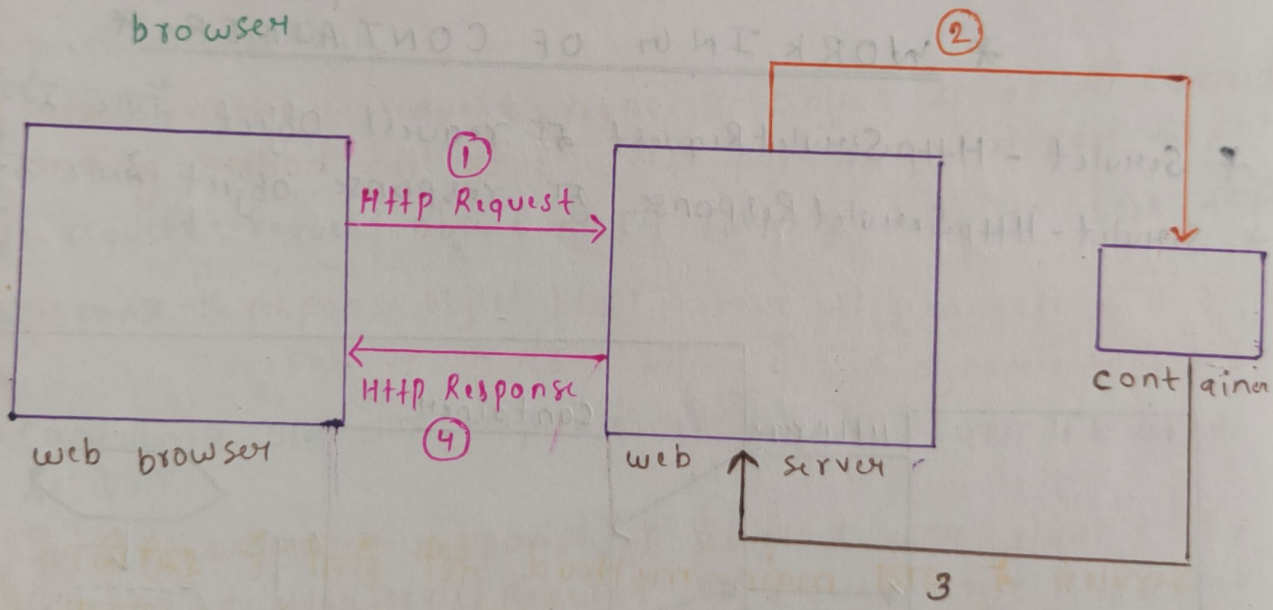
iv) Servlet के पास main method नही होती है इसीलिए Servlet को run करने का काम container करता है।

v) .class file container के पास जाएगी container उस .class file को execute करेगा और html output को generate करेगा। और वो html output browser के पास forword ho jayega।



Container :- Server के specific part को ही container कहते हैं।
container Server के ही अंदर रखा हुआ होता है।

जैसे Apache - Tomcat :- Apache Server का name है। और Tomcat उस server में रखा container कहलाता है।



1. सबसे पहले web browser से http request web server के पास गई

ii) 2 type of request
   1. html page के लिए
   2. Servlet page के लिए

iii) यदि html page के लिए request हुई तो लेकिन इस समय

request normal html page के लिए न होते हुए servlet के लिए है।

2. Server ने इस बार notice किया कि request Servlet के लिए है इसीलिए उसने इस बार request को container के पास forward कर दिया।

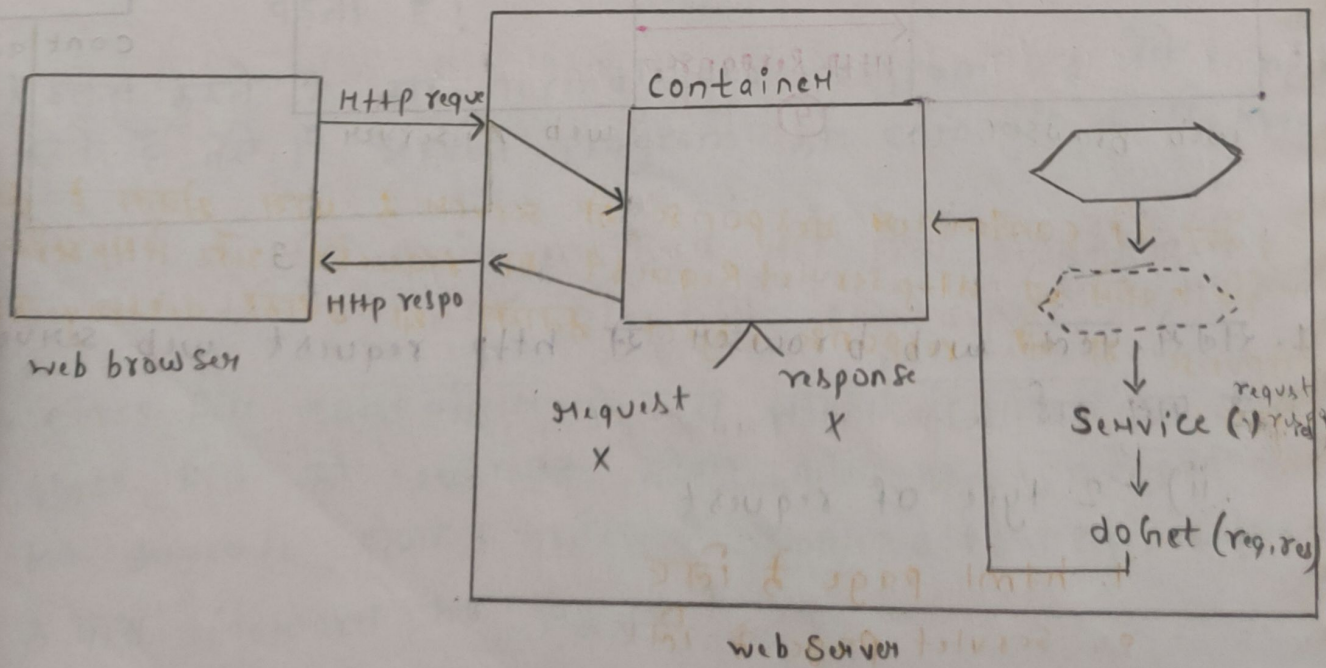
3. Container के पास अब request आ चुकी है और यहाँ बहुत सारी Servlet request हैं अपने पास तो जिस specific Container के corresponding request और है container उसको load करेगा और html. output कर देगा

4. Container उस html output को server के पास forward कर देगा

5. अब Server उस html output ko http response के form में web browser के पास forward kar dega

* WORKING OF CONTAINER *

* Servlet - HttpServletRequest का request object तैयार करेगे
* Servlet - HttpServletResponse का response object तैयार करेगे



1. Container सबसे पहले Http Request Server के पास भेजेगा फिर Servlet HttpServletRequest का request और HttpServletResponse का response object बनाकर तैयार करेगा

2] Container जिस servlet के लिए request आई है उस servlet की एक duplicate copy (thread) बनाकर तैयार कर देगा

3] Container अब उस thread के corresponding service method को call karega.

4] Container में जो HttpServletRequest का request और HttpServletResponse का response object बनकर तैयार हुआ है, उस समय request, response को service method के parameter में pass कर देगा (servlet).

5]. Service Method check करेगी उसके पास जो request आई है वो getMethod के through आई है या फिर post method के through aai hai!

6] यदि request doGet method से आई है तो internally doGet method call होगी और dopost method से आई है तो doPost method callhogi.

7] Service method request, response object को जो भी corresponding method call हुई है उसके parameter में pass कर देगी

8] i. request - request object से हम form के data को access करते

ii. Response -: response object html output को generate करने के purpose से use किया जाता है (in dynamic webpage)

9] Container अब उस response को server के पास forward कर देगा

10) जैसे ही container Response को server के पास भेजता है वैसे ही उसने जो HttpServletRequest का request और HttpServlet Response का जो response object बनाया था उसको destroy कर देगा।

11] Server उस http response के form में web browser के पास forward कर देता है।

07/09/23
Thrusday —

* JAVA SCRIPT [JS] *

javaScript -: JavaScript is a Scripting or programming language that allows you to implement complex feature on web-pages every time a web pages does more than just sit there and display static information for you to look at displaying timely content update, interactive maps, animated 2D/3D graphic scrolling video jukeboxes etc. you can bet that javaScript is probably involved. it is the third layer of the layer cake of standard web technologies two of which (HTML and CSS) we have covered in much more detail in other parts of the learning Area.

JavaScript को validation के purpose से use किया जाता है। जैसे कोई textfield है हमे उसमे 10 digit phone no ही store karna hai to या कोई textfield है उसमें capital char small char digit और special characters सब हो तभी valid hoga इस purpose से हम JS का use करते है।

Q जब हम java से validation कर सकते हो javaScript कि जरूरत क्यों पड़ी ?

Java के पास validation apply करने की पूरी facility है लेकिन जितनी भी बार हम गलत data enter करेंगे तो data server के पास Http request के form में जाएगा और server उसे Http response के form मे वापस browser के पास भेजेगा। और कई बार network issue के कारण time भी ज्यादा लग सकता है। बहुत lengthy process है इसीलिए javaScript का use किया है।

JavaScript में validation की जो process है वो

browsem मे ही हो जाएगी html जैसे servem मे जाने की कोई जरूरत नही पड़ेगी इसी purpose से javascript का use किया है।

JavaScript = Client side scripting language.

Java - Servem side scripting language.

i) Java का use :- जो हमें servem पर check करना है username password वहाँ validation के लिए java का use होगा।

username	password
aaa	111

ii) Js ka use :- Common validation के लिए javascript का use hoga like con.no 10digit

email - 12@na.com . Javascript के code को कभी .html extension से save करना पड़ेगा।

** "JavaScript is a scripting language that enables you to create dynamically updating content control multimedia, animated images, and pretty much everything else."

08/09/23
Friday

How to print a message from js :-

(1)
```
<html>
<body>
<script type="text/javascript">
document.write("Ram"); → by js
</script>
</body>
</html>
```
o/p- Ram

script को use हम <head> वाले part में भी कर सकते

(2)
```
<html>
<head>
<script type="text/javascript">
document.write("Ram");
</script>
</head>
</html>
```
o/p- ram

- js में airthmetic operator भी proper work करते हैं

(3)
```
<html>
<head>
<script type="text/javascript">
var x=10;
var y=20;
document.write("sum="+(x+y)+"<br>");
document.write("sub="+(x-y)+"<br>");
document.write("multi="+(x*y)+"<br>");
document.write("div="+(x/y)+"<br>");
document.write("mod="+(x%y)+"<br>");
```

+
-
*
/
%

```
</script>
</html>
</head>
```

```
Sum = 30
sub = -10
multi = 200
div = 0.5
mod = 10
```

यदि js में हमें किसी variable को display कराना है तो:

④
```
<html>
 <head>
<script type = "text/javascript">
var x=10;
var y = 10.6;
var z = "ram";
var z1 = false;
var z2='r';
document.write(x+"<br>");
document.write(y+"<br>");
document.write(z+"<br>");
document.write(z1+"<br>");
document.write(z2+"<br>");
</script>
</head>
</html>
```

O/P - 10
10.6
ram
false
r

NOTE -:. इस var में all type का data display करा सकते है, int, char, float, boolean, string, etc.

js में all type के relational operator भी work करते है।

⑤
```
<html>
<head>
<script type = "text/javascript">
```

```
var x=10;
var y=20;
document.write((x<y)+"<br>");
document.write((x>y)+"<br>");
</script>
</head>
</html>
```

o/p - true
false

<
>
<=
>=
==
!=

increment / decrement भी proper work करेगा

⑥
```
<html>
<head>
<script type="text/javascript">
var x=10;
x++;
document.write(x+"<br>");
</script>
</html>
</head>
```

o/p - 11

js में assignment operator भी proper work करता

⑦
```
<html>
<head>
<script type="text/javascript">
var x=10;
var y=20;
x=y;
document.write(x);
</script>
</head>
</html>
```

o/p = 30

js में all type के bitwise operator भी work karte hai

&	^	<<
|	~	>>

js में program control statement भी work करते है

⑧
```
<html>
<head>
<script type="text/javascript">
var x=10;
var y=20;
if(x>y)
{
document.write("x is largest");
}
else
{
document.write("y is largest");
}
</script>
</head>
</html>
```

else if, else if ladder, switch case का भी use हम js में कर सकते है

* All type of loops का use भी हम js में कर सकते है।

⑨
```
<html>
<head>
<script type="text/javascript">
for(i=1; i<=10; i++)
```

```
for(i=1; i<=10; i++)
{
document.write("ram <br>");
}
</script>
</head>
</html>
```

js मे हम function को बनाकर भी तैयार कर सकते है,

(10)
```
<html>
<head>
<script type="text/javascript">
function demo()
{
document.write("ram","<br>");
}
demo();
demo();
demo();
</script>
</head>
</head>
```

o/p - ram
ram
ram

js मे parameterized function भी बना सकते है!

(11)
```
<html>
<head>
<script type="text/javascript">
function demo(x,y)
{
document.write("sum="+(x+y));
}
demo(10,20);
</script>
</html>
</head>
```

o/p - sum=30

NOTE :- यदि हमे js मे function बनाना है तो उसका return type function ही देना होगा क्योकि js में कुछ और return type नही होता है।

js मे return type कुछ होता ही नही है तो specify करने की जरूरत नही है

(12)

```
<html>
<head>
<script type="text/javascript">
function demo(x,y)
{
return x+y;
}
document.write(demo(10,20));
</script>
</head>
</html>
```

o/p - 30

working OR || operator

(13)

```
<html>
<head>
<script type="text/javascript">
function demo()
{
document.write("ram <br>");
return true;
}
function demo2()
{
document.write("sita <br>");
return false;
}
if( demo() || demo2() )
{
document.write("welcome <br>");
```

→ यदि यहाँ false return किया तो ram sita invalid data aayega.

```
}
else
{
document.write("invalid data <br>");
}
</script>
</head>
</html>
```

o/p - ram
welcome

(14)

```
<html>
<head>
<script type="text/javascript">
function demo()
{
document.write("ram <br>");
return true;
}
function demo2()
{
document.write("sita <br>")
return false;
}
if(demo() && demo2())
{
document.write("welcome <br>");
}
else
{
document.write("invalid data <br>");
}
</script>
</head>
</html>
```

o/p - ram
sita
invalid data

(15)

```
<html>
<head>
<script type="text/javascript">
 function demo()
 {
 document.write("ram <br>");
 }
demo();
</script>
</head>
<body>
<p> sita </p>
</body>
</html>
```

o/p -

ram

sita

(16)

```
<html>
<head>
<script type="text/javascript">
function demo()
{
document.write("ram <br>");
}
demo();
</script>
</head>
<body onload="demo()">
<p> sita </p>
</body>
</html>
```

o/p - ram

```
<html>
<head>
<script type="text/javascript">
function demo()
{
document.getElementById('x').innerHTML='ram';
}
demo()
</script>
</head>
<body onload="demo()">
<p> sita </p>
<p id="x"> </p>
<p> gita </p>
</body>
</html>
```

o/p- sita
ram
gita

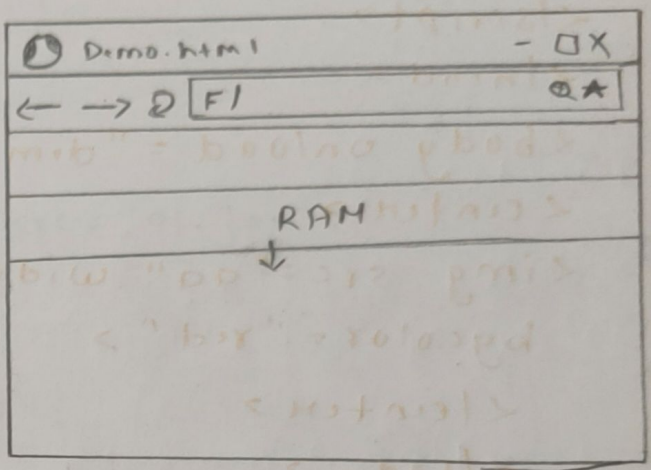
09/09/23
saturday

1.
```
<html>
<head>
<script type="text/javascript">
function demo()
{
document.write("ram <br>");
setTimeout(demo,1000);
}
</script>
</head>
<body onload="demo()">
</body>
</html>
```

ram
ram
ram
ram
:
:

②
```
<html>
<head>
<script type="text/javascript">
c=1;
function demo()
{
if(c==1)
{
document.getElementById('x').innerHTML="RAM";
document.getElement By Id('x').style background="black";
document.getElement By Id('x').stylecolor="white";
c++;
}
else if(c==2)
{
document.getElement By Id('x').innerHTML="SITA";
document.getElement By Id('x').style.background="red";
document.getElement By Id('x').style.color="white";
c++;
}
```

```
else
 {
document.getElementById('x').innerHTML = "GITA";
document.getElementById('x').style.background = "green";
document.getElementById('x').style.color = "white";
 c=1;
}
setTimeout(demo, 1000);
 }
</script>
 </head>
<body onload="demo()">
 <center>
 <h1 id='x' ></h1>
  </center>
  <body>
  </html>
```

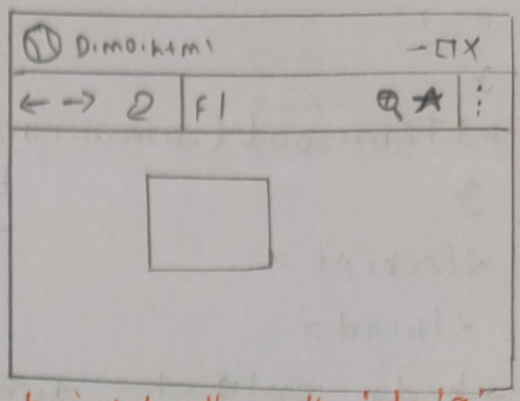


③

```
<html>
 <head>
 <script type = "text/javascript">
  c=1;
 function demo()
  {
  switch (c)
   {
    case 1:
document.getElementById('x').src = "a1.jpg";
  c++;
  break;
 case 2:
document.getElementById('x').src = "a2.jpg";
 c++;
 break;
```

```
case 3:
document.getElementById('x').src="a3.jpg";
c=1;
break;
}
setTimeout(demo,1000);
}
</script>
</head>
<body onload="demo()">
<center>
<img src="aa" width="1000" height="700" id='x'
bgcolor="red">
</center>
</body>
</html>
```

img पर एक second में change hogi



④

```
<html>
<head>
<script type="text/javascript">
c=1;
function demo()
{
document.getElementById('x').src="a2.jpg";
}
function demo()
{
document.getElementById('x').src="a1.jpg";
}
</script>
</head>
<body onload="demo()">
<center>
<img src="aa" width="300" height="300" id='x'
```

```
onmouseover="demo2()">
</center>
</body>
</html>
```

⑤

```
<html>
<head>
<script type="text/javascript">
c=1;
function demo()
{
document.getElementById('x').src="a1.jpg";
}
</script>
</head>
<body>
<center>
<img src="a3.jpg" width="300" height="300" id="x">
<input type="button" value="click" onclick="demo()">
</center>
</body>
</html>
```

⑥

```
<html>
<head>
<script type="text/javascript">
c=1;
function demo()
{
document.getElementById('t1').style.blackground="black";
document.getElementById('t1').style.color="white";
}
function demo2()
{
document.getElement By Id('t2').style.background="black";
document.getElement By Id('t2').style.style.color="white";
```

```
}
  function demo3()
  {
document.getElementById('t1').style.background="white";
document.getElementById('t1').style.color="black";
  }
  function demo4()
  {
 document.getElementById('t2'). style.background="white";
document.getElementById('t2').style.background='black';
  }
function demo5()
 {
document.getElementById('t1').style.background="red";
document.getElementById('t2'). style.color="white";
  }
  function demo6()
  {
  document.getElementById('t2'). stylebackground="blue";
document.getElementById('t2').style.color="white";
  }
  function demo7()
  {
 document.getElementById('t1'). style.background="red";
document.getElementById('t1'). style.background~~color~~="white";
  }
  function demo8()
  {
document.getElementById('t2'). style.background="blue";
document.getElementById('t2'). style.color="black";
  }
  </script>
  </head>
  <body>
  <center>
<input type="text" id="t1" onclick="demo()" onblur="demo3()"
      onkeyup="demo5()" onkeydown="demo6()">
```

```
<input type="text" id="t2" onclick="demo2()" onblur
   ="demo4()" onkeyup="demo7()" onkeydown="demo5()">
<h1 id='x'></h1>
</center>
</body>
</html>
```

(7)

```
<html>
<head>
<script type="text/javascript">
 c=1;
function demo(a)
{
document.getElementById('t1').style.background="black";
document.getElementById('t1').style.color="white";
}
function demo3(a)
{
document.getElementById('t2').style.background="blue";
document.getElementById('t2').style.color="white";
}
function demo5(a)
{
document.getElementById('t1').style.background="black";
document.getElementById('t1').style.color="white";
}
function demo6(a)
{
document.getElementById('t2').style.background="blue";
document.getElementById('t2').style.color="white";
</script>
</head>
<body>
<center>
```

```
<input type = "text" id = "t1" onclick = "demo (this)"
onblur = "demo3(this") Onkeyup = "demo5(this)" >
<input type = "text" id = "t2" onclick = "demo(this)" onblur
= "demo3(this)" onkeyup = " demo5(this)" onkeyup = "demo
6(this)" >
</center>
</body>
</html>
```

* alert box

⑧ abc.js

```
function demo()
{
alert("welcome");
}
```

demo.html

```
<html>
<head>
<script type = "text/javascript" src = "abc.js">
</script>
</head>
<body>
<center>
<input type = "button" value = "click" onclick = "demo()">
</center>
</body>
</html>
```

⑨ किसी भी message को delete करने पर conformation box आना चाहिए

ab.js

⑨

```
function demo()
{
x=confirm("r u sure");
 if(x)
 {
 alert("data delete");
 }
 else
 {
alert("data not delete");
 }
}
```

Demo.html

```
<head>
<html>
<script type="text/javascript" src="ab.js">
</script>
</head>
<body>
<center>
<input type="button" value="click" onclick="demo()">
</center>
</body>
</html>
```

dialog box

⑩ ab.js

```
function demo()
 {
 y=prompt("Enter Name");
document.getElementById('x').innerHTML=y;
```

Demo.html

```
<html>
<head>
<script type="text/javascript" src="ab.js">
</script>
</head>
<body>
<center>
<input type="button" value="click" onclick="demo()">
<h1 id="x"></h1>
</center>
</body>
</html>
```

(11)

ab.js

```
function demo()
{
x= document.getElementById('t1').value;
document.getElementById('t2').value=x;
document.getElementById('t1').value="";
}
```

Demo.html

```
<html>
<head>
<script type="text/javascript" src="db.js">
</script>
</head>
<body>
<center>
<input type="text" id="t1">
<input type="button" value="click" onclick="demo()">
<input type="text" id="t2">
</center>
</body>
</html>
```

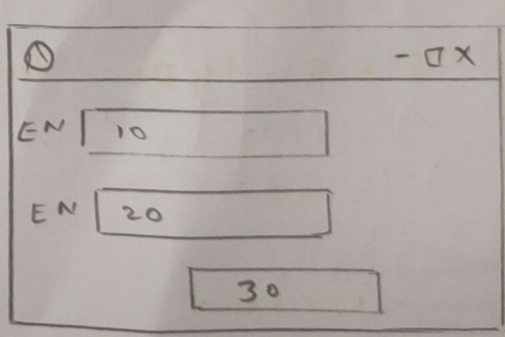
(12) 2 NO ka sum ka program

ab.js

```
function demo()
{
x= document.getElementById('t1').value;
y=document.getElementById('t2').value;
 a=parseInt(x);
 y=parseInt(y);
document.getElementById('t3').value=(a+b);
}
```

demo.html

```
<html>
<head>
<script type="text/javascript" src="ab.js">
</script>
<head>
<body>
<center>
Enter NO 1 <input type="text" id="t1"><br>
Enter NO 2 <input type="text" id="t2"><br>
<input type="button" value="click" onclick="demo()">
<input type="text" id="t3">
</center>
</body>
</html>
```



(13)

```
function demo()
{
y=document.getElemantById('t1').value;
if(isNaN(y))
```

```
{
document.getElementById('x').innerHTML = "Not a NO";
}
else
{
document.getElementById('x').innerHTML = " is A NO";
}
}
```

Demo.html

```
<head>
<script type = "text/javascript " src = "ab.js">
</script>
</head>
<body>
<center>
<input type = "text" id = "t1" > <br>
<input type = " button" value = "click" onclick = "demo()">
<h1 id = "x"> </h1>
</center>
</body>
</html>
```

9. Table ko check karne ke liye

mysql > desc table name

10. Table ka data delete karne ke liye

mysql > delete from sita;

11. Table ko show karne ke liye;

mysql > show table

12. Table ka particular data delete karne ke liye;

mysql> delete from geeta where uname='ram';

13. Table ka data update karne ke liye.

mysql > UPDATE geeta SET UCN = 922 WHERE UNAME = 'aa'

UPDATE table_name SET column_name where Name=' ';

* MY SQL Query *

1. Database ko show karne ke liye
   :- mysql> show databases;

2. Database me se particular database ko use karne ke Liye.
   :- mysql > use database databases name;

3. Database me table create karne keliye;
   :- mysql> create table table name (UNAME varchar (30), UPASS varchar (30));
   > create table ramji (UNAME varchar(30), UPASS varchar(30));

4. Table me data insert karne ke liye
   mysql > insert into table name values ('values', 'values');
   > insert into ramji values ('ra', '123');

5. Table ka data show karne ke liye
   mysql > select * form table-name;

6. * ka matlab all Row show karna hota he

Table ka particular coloumn show karne ke liye
mysql> select coloumn name from table name
> select UNAME from ramji;

Table me particular row ka data show karne ke liye
mysql> select *from table name where coloumn name = 'data';
> select *from ramji : where UNAME = 'aaa';