**datatype conversion:**

1. Integer(non prim)=Integer.valueOf(intval);
2. int(primitive)=Integerdata.intValue();
3. (int)12.5
4. final int KEY=90 (const)
5. arry to list: list =Array.asList(arr); or Collections.addAll(list, arr);
6. list to array : array= listname.toArray(new String[0]);

**Props:**

1. array=.length
2. string=.length()
3. non primitive (like lists)=size();
4. .equals() checks for value quality only,
5. == checks both value and memory equality if obj compared with obj.(but in string only value compares)
6. Scanner sc=new Scanner(System.in)
7. String inp=sc.nextLine()
8. Math.random() (0-1)
9. int[] arr=new int[] or int[] arr={1,2,3,4,7} /// Array.sort(arr),arr.length
10. String str="gnjdjgd" str.length() ,str.substring(start[],end())

**OOPS:**

1. Instance Initializer block : to Initialize data members,
2. Final methods cant be overrride
3. final var is like const,if blank,then can be initialised only in construters ,
4. final class cant be extended/inherited
5. final params cant be changed (params++ not allowed)
6. final static blank vars can be changed only in static block
7. If there is any private, final or static method in a class, there is static binding.

**STATIC:**

1. static method cannot access not static methods/data members without referencing/making obj them , but only static ones but inverse is true.
2. bcoz static methods are instances/object independant,so u cannot access some instances dependent methods.
3. static variables can be initialised only in static block (which runs only once before main() as well).
4. All static stuffs are resolved during compile time only, as they are independant of objects which are generated at run time
5. static methods and instance variables can be inherited but not overridden as they belongs to parent class only no matter who all are using it.

**INHERITENCE:**

1. parent class pointers can point/store base classes address/references but they cannot access childs properties
2. super(a,b,c) calling direct parent constructor
3. we can pass base object in super as super(baseObj);

**ENCAPSULATION:**

* private :accessible in current class only
* default :accessible in current package only
* protected:accessible in inheriting child classes and pakg
* public:accessible everywhere

**POLYMORPHISM:**

1. compiled,runtime
2. upcasting: base ptr refering child obj method

**COLLECTIONS:**

1. Packages:userdefined and built ins=[lang,io,utils and collections,applet(spring boot),awt(for GUI),net(networks)]
2. Object class is top most class(inherited by default indirectly)
3. finalise is called while garbage collector comes to clean the objects

**ABSTRACT CLASS :**

1. (having atleast 1 abstract methods,method without body)
2. in order to address multiple inheritence issue, it is devised.
3. class having atleast one Abstract method(methods with no body) must be abs class so write it in class decleration as well.
4. All abs methods has to be defined in the base classes who all are inheriting it.
5. parent abs class cannot be instanciated directly,but by overriding its abs methods
6. abs constructor cannot be created

**INTERFACES:**

1. (having only abs methods,and static or default methods)
2. we gave a soln of multiple inheritence issue, but cannot make multiple inheritence,so interfaces comes into picture.
3. just use implements instead of extends in child class
4. static method in interface and abs class should always be implemented as they cannot be overriden by child implementors
5. overriden methods should be more accessible wrt parent access modifiers(e.g.=protected in parent but pub in childs)

**NON OOPS**

1. datatypes[] variable\_name=new datatype[size]/{1,2,3}
2. Functional interface:(interface with only 1 abstract method).lots of inbuilt fun int. are there in java 1.8;
3. Lambda Expressions:same as arrow function of js. These are used to define functional interfaces method;

**Servlet:**

1. req.setAttributes("name",val)
2. req.requestDisaptcher(): to call nested servlets in same domain;
3. res.sendRedirect("newServlet?params=67657"); //url rewriting method
4. httpSession ssn=req.GetSession();
5. ssn.setAttribute("key",val); can be recieved in new servlet by creating session obj and getattribute method like localStorage;
6. servletcontext():u can create global key,values and can access it from any servlets;
7. servletconfig():inside particular servlet we can specify key,values which are accessible by that particular servlet only;
8. we can use annotion method to avoid web.xml method:
   1. @WebServlet("/url") above the defined servlet function

**JSP:**

1. its like jsx(convert html like code to java);

<html>

<%

@page used for import (called directive tag)

%>

<%

for global class and variable declaration (called declaration tag)

%>

<%

scriptlet(go to service)

%>

<%=

used for out (direct printing called expression tag)

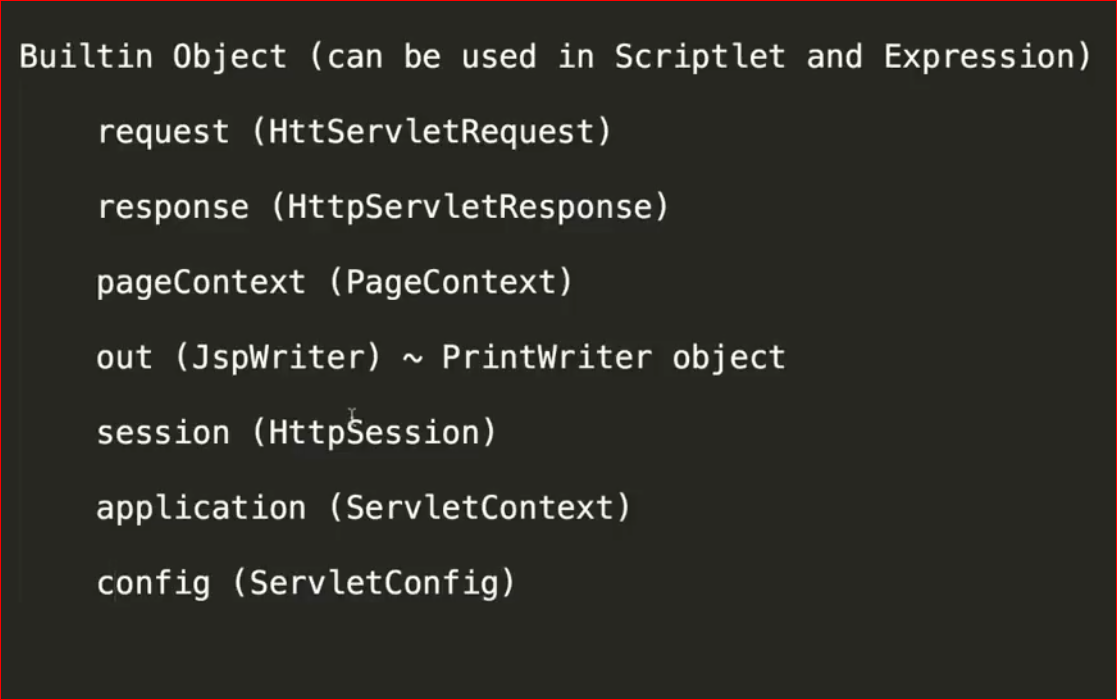
%>

</html>

**JSP DIRECTIVE:**

1. <@page: for importing libraries
2. <@include: to include local file classes
3. <@Taglib:we can import user defined tags

There are few builtin objects which can be used without instantiating them:



Using pagecontext we can set key value pairs and can modify its scope.

MVC

