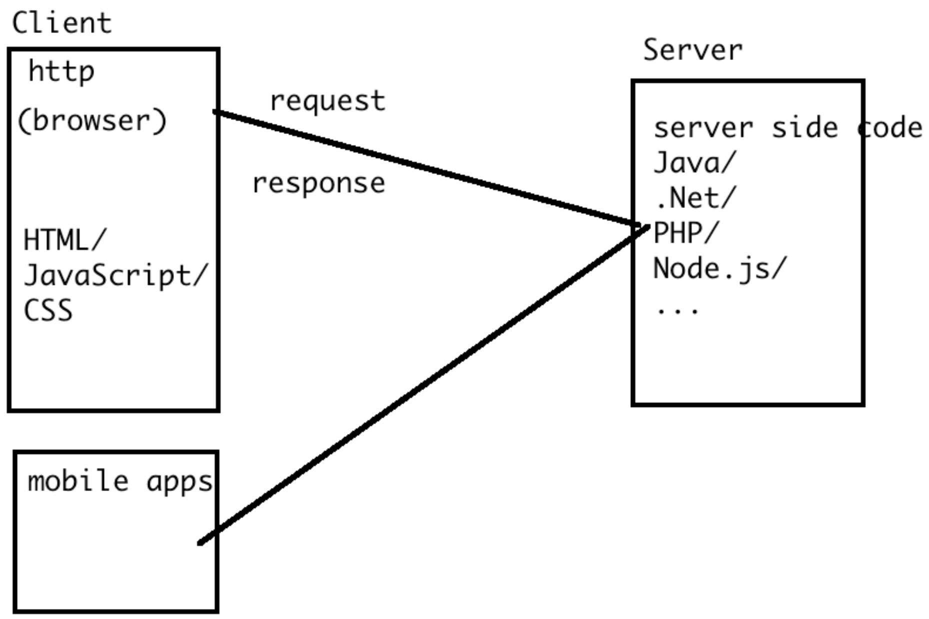
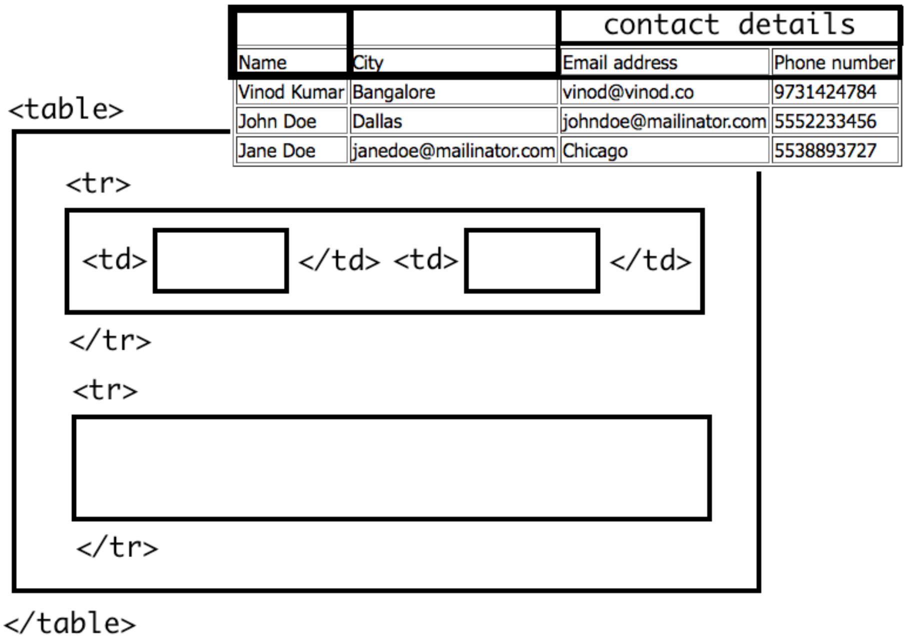
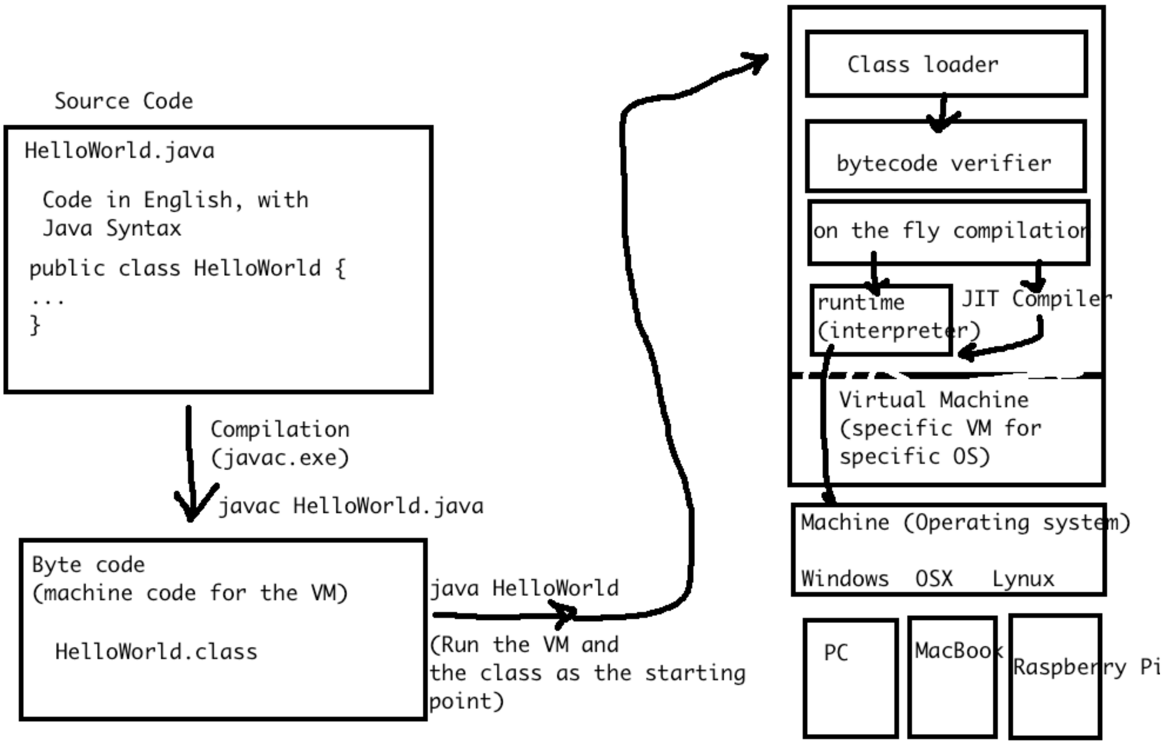
-


Client
http
(browser)
HTML/
JavaScript/
CSS
mobile apps
request
response
Server
server side code
Java/
.Net/
PHP/
Node.js/
...



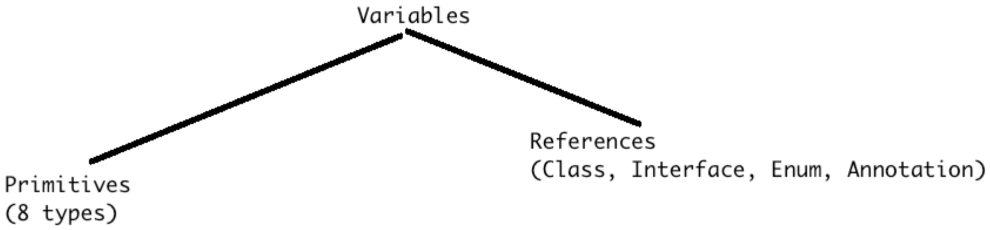
		contact details	
Name	City	Email address	Phone number
Vinod Kumar	Bangalore	vinod@vinod.co	9731424784
John Doe	Dallas	johndoe@mailinator.com	5552233456
Jane Doe	janedoe@mailinator.com	Chicago	5538893727



```
public class HelloWorld {
...
}
```

```
javac HelloWorld.java
```

```
java HelloWorld
```



Variables

Primitives
(8 types)

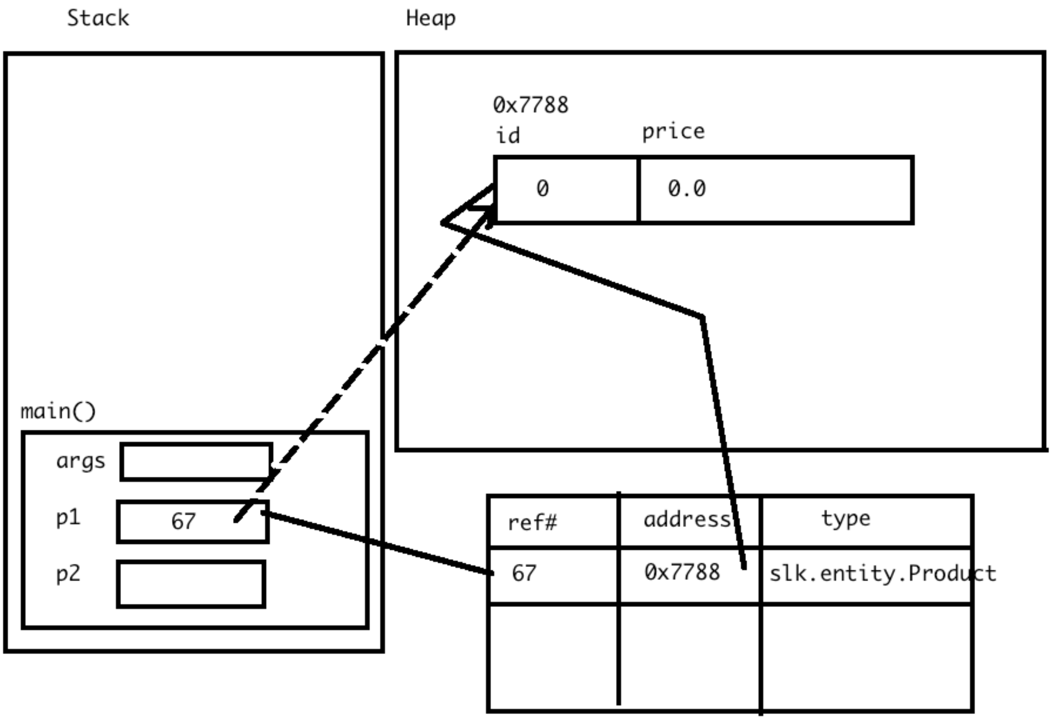
References
(Class, Interface, Enum, Annotation)

1. Integers:
   byte (1 byte)
   short (2 bytes)
   int (4 bytes)
   long (8 bytes)

2. Real numbers
   float (4 bytes)
   double (8 bytes)

3. Characters
   char (2 bytes)

4. Boolean / logical
   boolean (1 bit)
   true/false



Stack

Heap

main()

args

p1

67

p2

0x7788

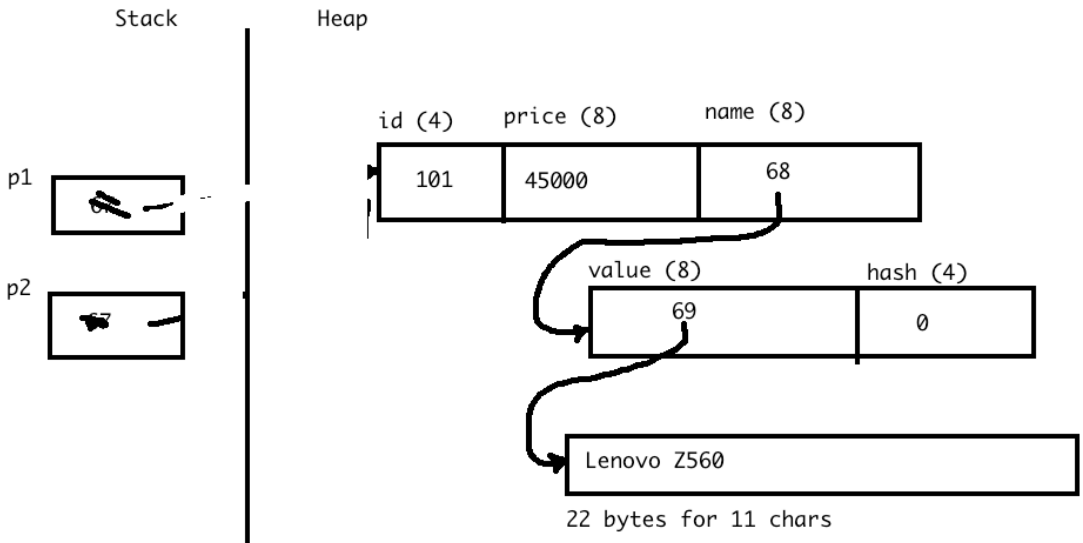
id

price

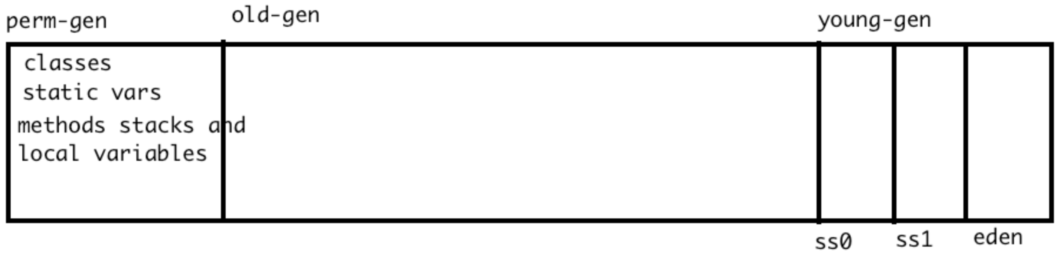
0

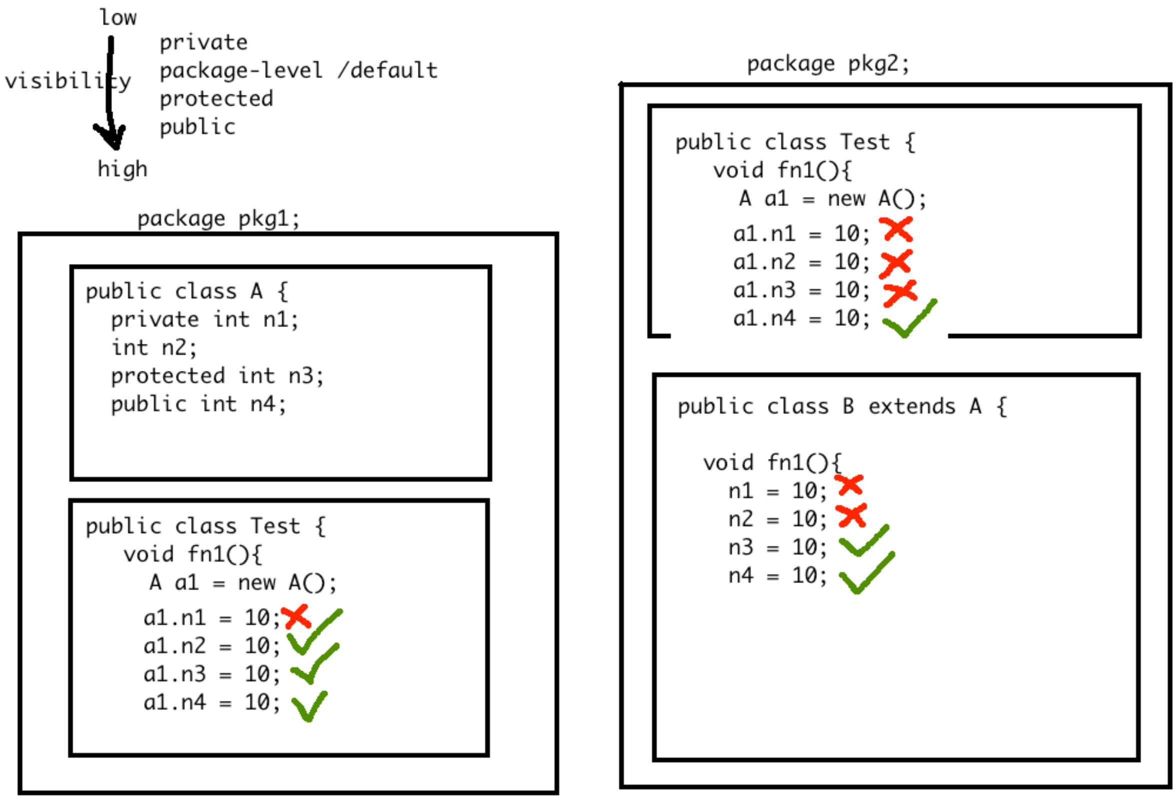
0.0

ref#	address	type
67	0x7788	slk.entity.Product



Initial memory allocation to the JVM is 1/64 th of the RAM
Maximum memory allocation to the JVM is 1/4 th of the RAM





```
private
package-level /default
protected
public
```

package pkg1;

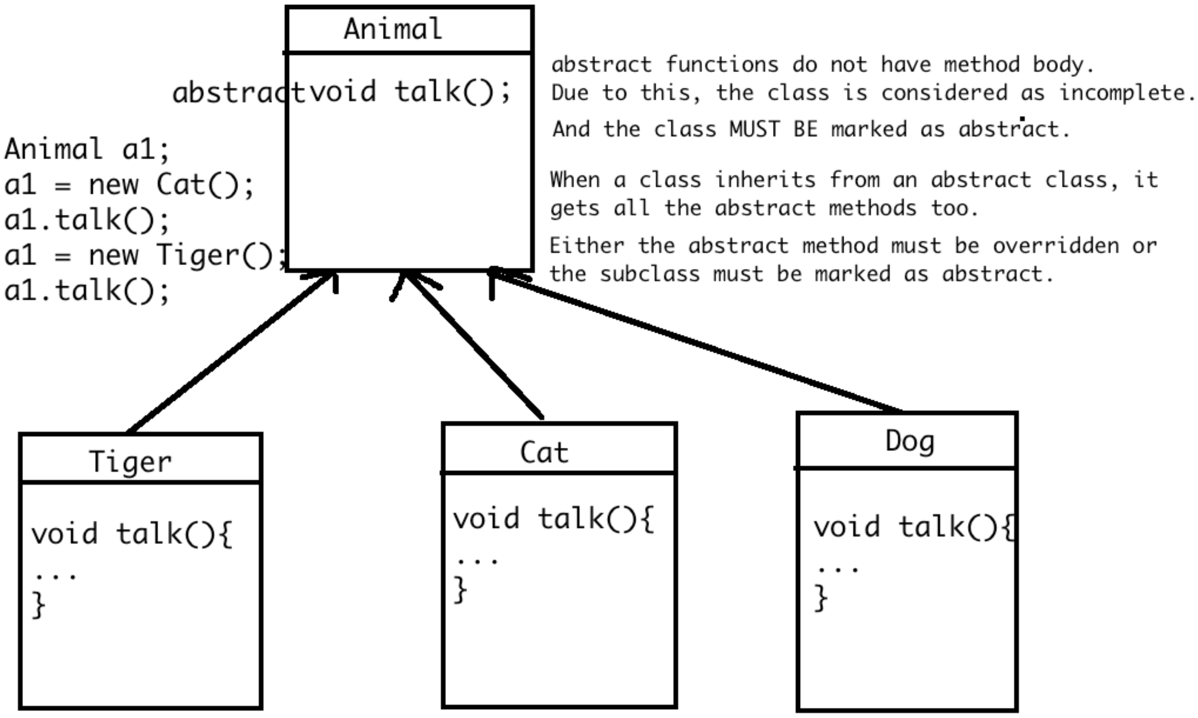
```
public class A {
  private int n1;
  int n2;
  protected int n3;
  public int n4;
```

```
public class Test {
   void fn1(){
     A a1 = new A();
    a1.n1 = 10; ✗
    a1.n2 = 10; ✓
    a1.n3 = 10; ✓
    a1.n4 = 10; ✓
```

package pkg2;

```
public class Test {
   void fn1(){
     A a1 = new A();
    a1.n1 = 10; ✗
    a1.n2 = 10; ✗
    a1.n3 = 10; ✗
    a1.n4 = 10; ✓
```

```
public class B extends A {

  void fn1(){
    n1 = 10; ✗
    n2 = 10; ✗
    n3 = 10; ✓
    n4 = 10; ✓
```



```
abstractvoid talk();
```

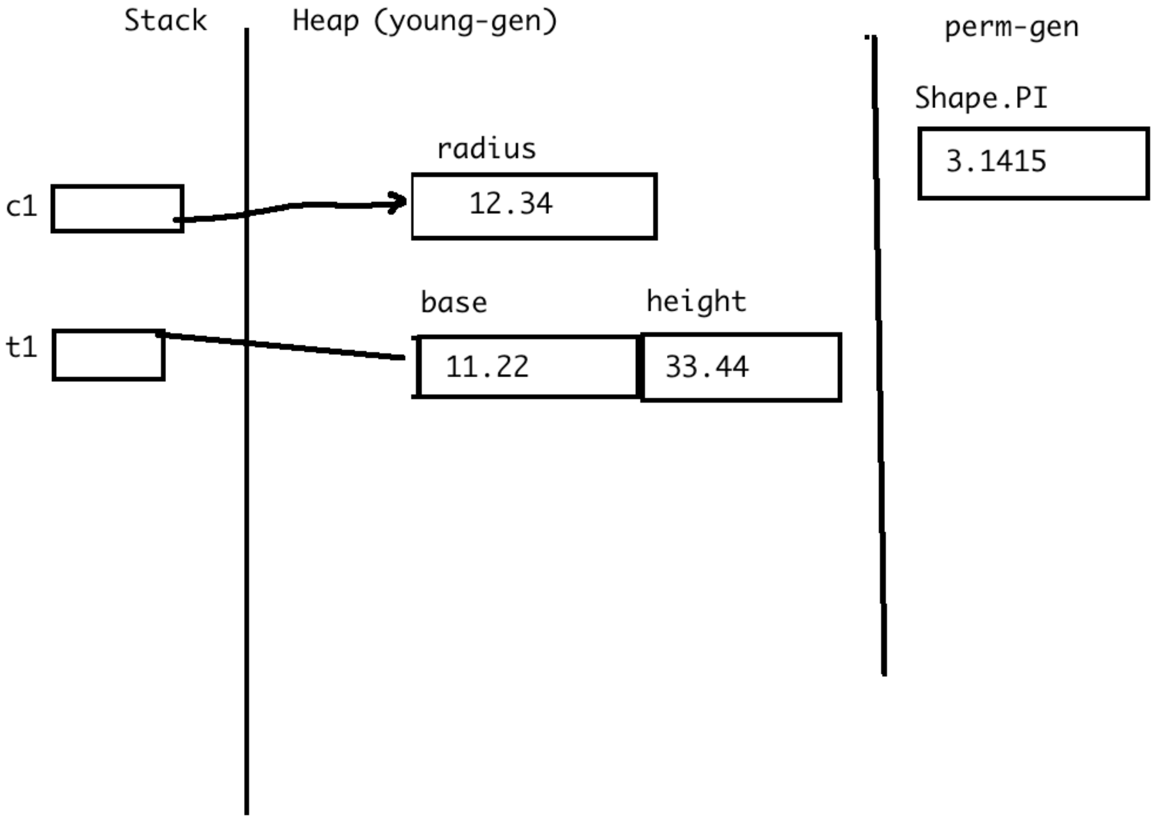
```
Animal a1;
a1 = new Cat();
a1.talk();
a1 = new Tiger();
a1.talk();
```

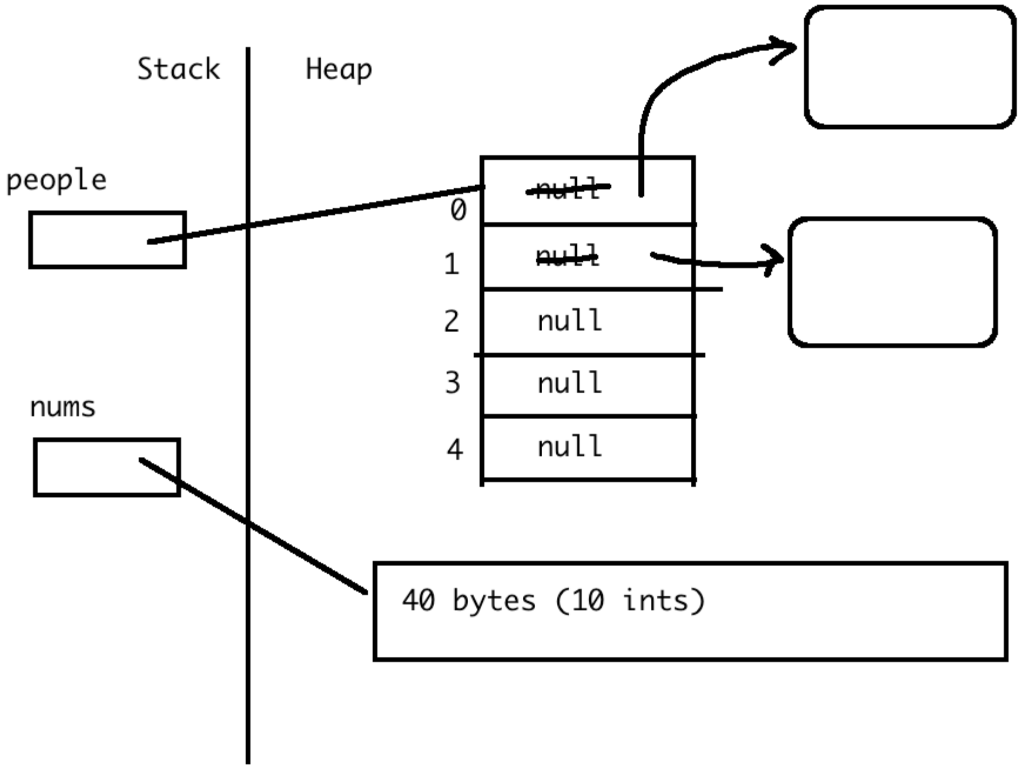
```
void talk(){
...
}
```

abstract functions do not have method body.
Due to this, the class is considered as incomplete.
And the class MUST BE marked as abstract.

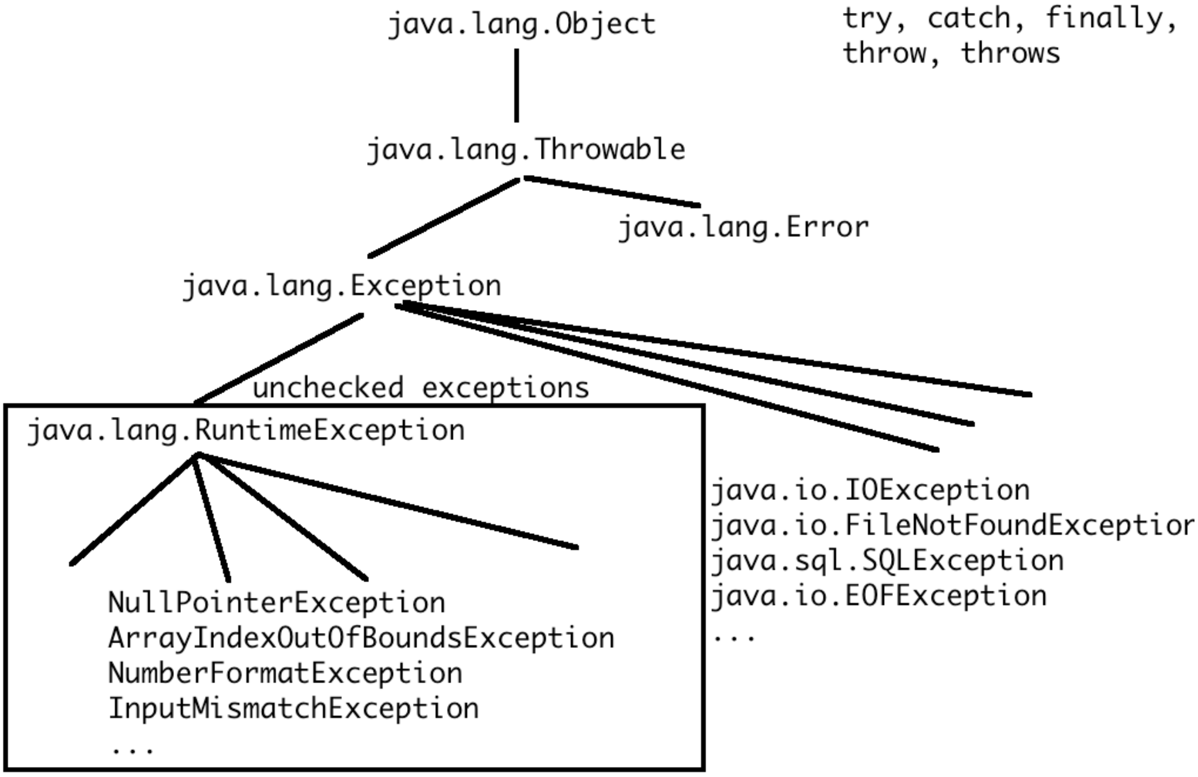
When a class inherits from an abstract class, it gets all the abstract methods too.

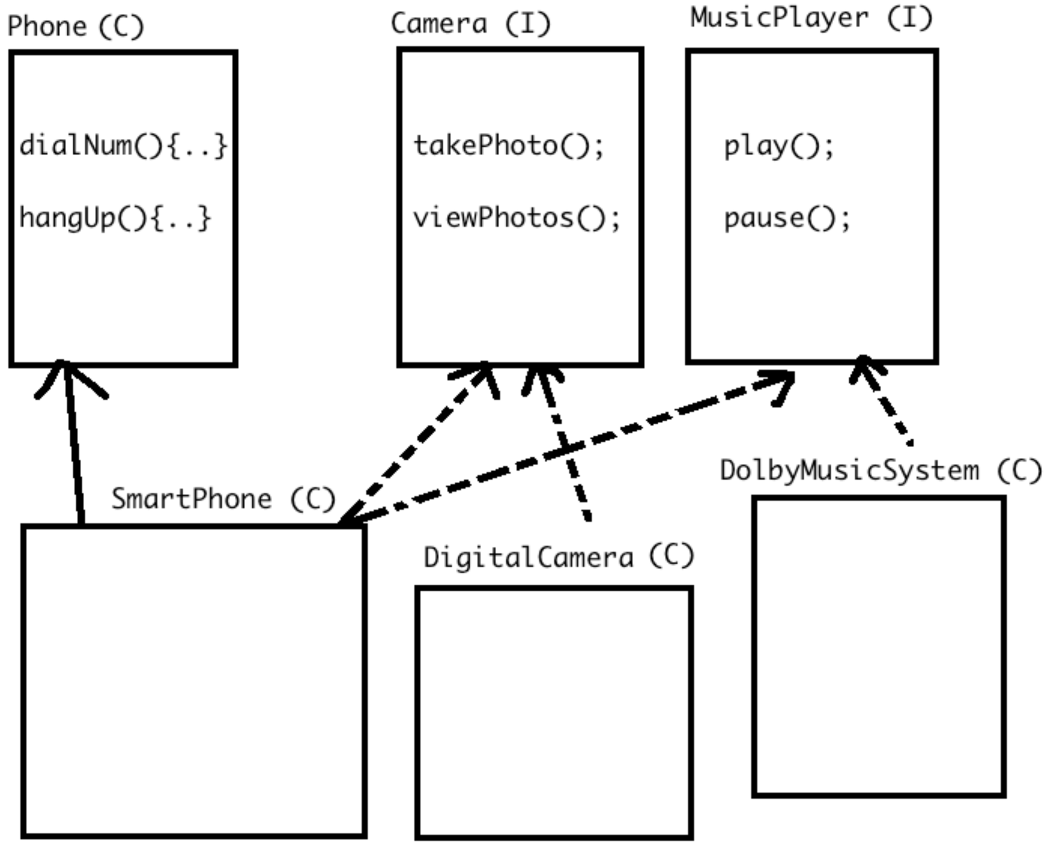
Either the abstract method must be overridden or the subclass must be marked as abstract.

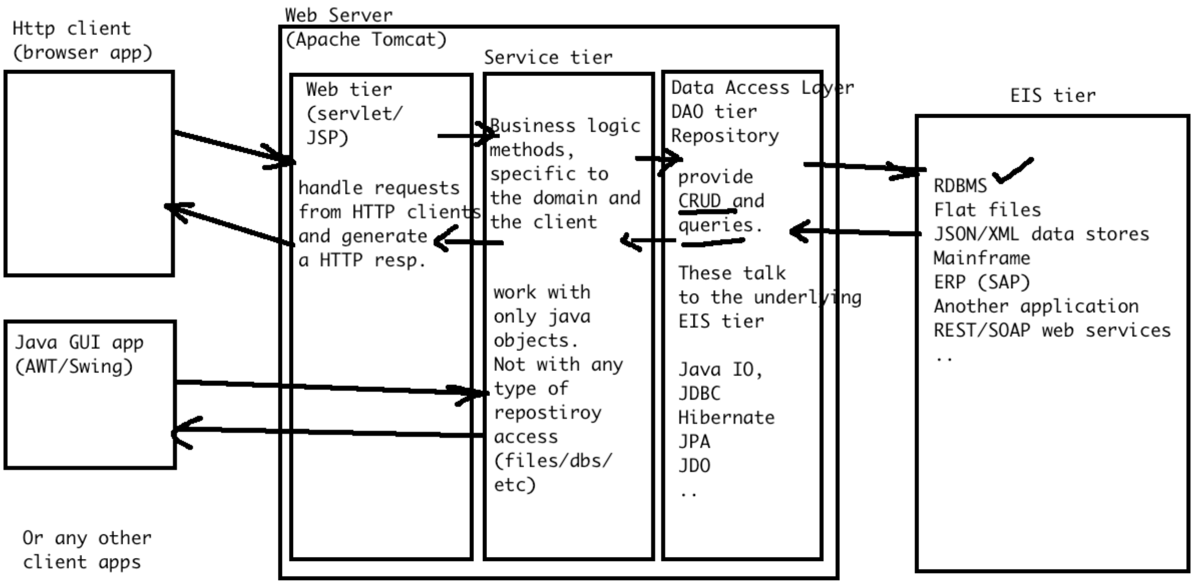

Stack
Heap (young-gen)
perm-gen
Shape.PI
3.1415
radius
c1
12.34
base
height
t1
11.22
33.44

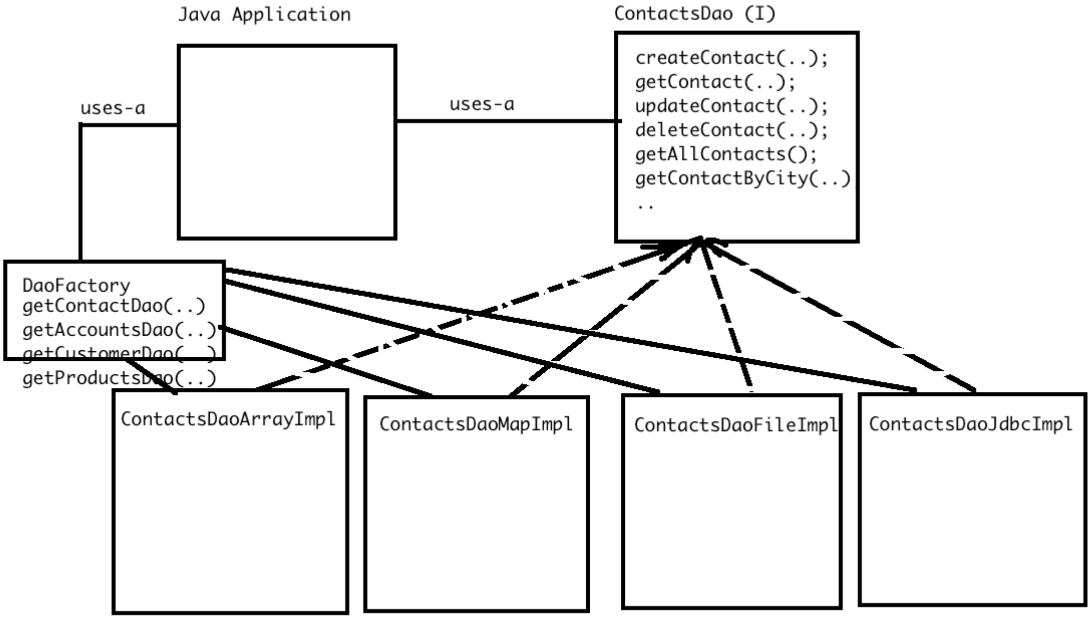

Stack
Heap
people
0
~~null~~
1
~~null~~
2
null
3
null
4
null
nums
40 bytes (10 ints)

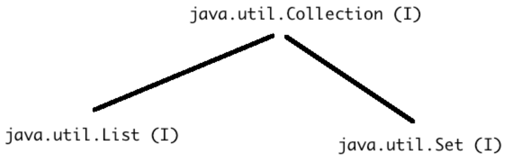
try, catch, finally,
throw, throws






Http client
(browser app)
Web Server
(Apache Tomcat)
Web tier
(servlet/
JSP)
handle requests
from HTTP clients
and generate
a HTTP resp.
Service tier
Business logic
methods,
specific to
the domain and
the client
work with
only java
objects.
Not with any
type of
repostiroy
access
(files/dbs/
etc)
Data Access Layer
DAO tier
Repository
provide
CRUD and
queries.
These talk
to the underlying
EIS tier
Java IO,
JDBC
Hibernate
JPA
JDO
..
EIS tier
RDBMS ✓
Flat files
JSON/XML data stores
Mainframe
ERP (SAP)
Another application
REST/SOAP web services
..
Java GUI app
(AWT/Swing)
Or any other
client apps


Java Application
uses-a
uses-a
ContactsDao (I)
createContact(..);
getContact(..);
updateContact(..);
deleteContact(..);
getAllContacts();
getContactByCity(..)
..
DaoFactory
getContactDao(..)
getAccountsDao(..)
getCustomerDao(..)
getProductsDao(..)
ContactsDaoArrayImpl
ContactsDaoMapImpl
ContactsDaoFileImpl
ContactsDaoJdbcImpl



java.util.Collection (I)

java.util.List (I)

* provides index based operations
* order of insertion is maintained
    while retrieval

Most important classes that
implement the List interface:

1. java.util.ArrayList

   * must be the default choice
   * insertion/deletion at random index
       is less performing

2. java.util.LinkedList

   * preferred when insertion/deletion
       is frequently done at random
       indexes.

3. java.util.Vector
   * legacy
   * many methods are synchronised, and hence
       is preferred in a multithreaded application
       where this collection is shared by many threads

java.util.Set (I)

* No additional methods provided
* No index based operations
* puts a restriction on implementing classes that
    there are no duplicate elements in the collection
* order of insertion is not guaranteed to be same as
    order of retrieval

Some of the most commonly used implementations:

1. HashSet
   * internally uses another collection called HashMap
   * the elements to be added to this collection must contain
       equals() and hashCode() functions
   * order of insertion and retrieval may differ
2. TreeSet
   * internally uses a red-black tree
   * the elements must implement java.lang.Comparable
   * the order of retrieval is in ascending order of the elements
3. LinkedHashSet
   * similar to HashSet
   * the order of retrieval is same as the order of insertion

java.util.Map (I)

* elements are stored and accessed using a "key"
  rather than a numerical index

* key-value collection

* key must be unique

Popular implementations:

1. java.util.HashMap

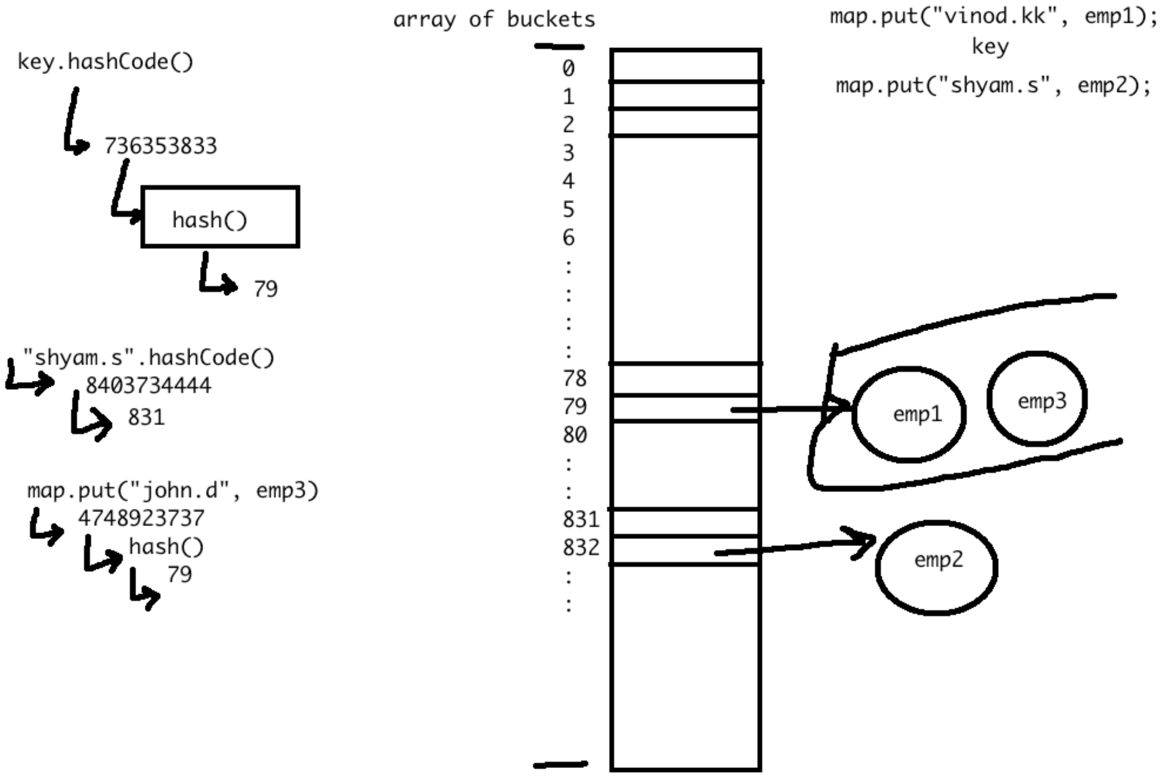
   * the key can be any object, but must provide hashCode() and equals()
   * If the key is duplicate or not is decided by the outcome of hashCode() and equals()
   * The object represented by the key has no restrictions
   * does not guarantee the order of retrieval to be same as order of insertion
   * if a key is repeated, then it replaces the previously stored value

2. java.util.TreeMap

   * the key can be any object, but must be a Comparable object
   * when the values are retrieved, they will be in ascending order of their keys

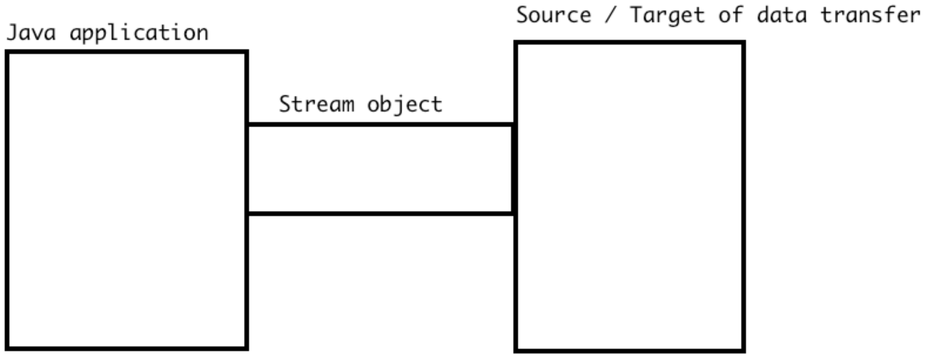
3. java.util.LinkedHashMap

   * The key can be any object that implements hashCode() and equals()
   * The order of insertion is maintained, and when you retrieve the values,
       they will be in the order as how they were added



```
java.util.Comparator<T>

public int compare(T t1, T t2)

* logic to compare t1 and t2

* return value must be -ve, zero or +ve

  if -ve, t1 < t2
  if +ve, t1 > t2
  if 0, t1 == t2

* used by many utility methods, such as:
  Collections.sort
  TreeMap, TreeSet constructors
  ...
```

Java Input/Output Streams



Java application

Stream object

Source / Target of data transfer

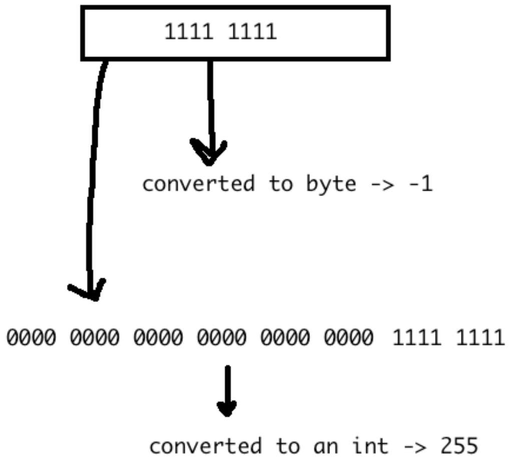
Console input
   (keyboard, mouse, scanner, barcode reader..)
Console output
   (Monitor, printer, speaker, ..)
File (text/binary)
Another Java application,
...

java.io.InputStream
abstract class

BufferedInputStream
DataInputStream
ObjectInputStream
} decorators

FileInputStream
SocketInputStream
ServletInputStream

java.io.OutputStream
abstract class



1111 1111

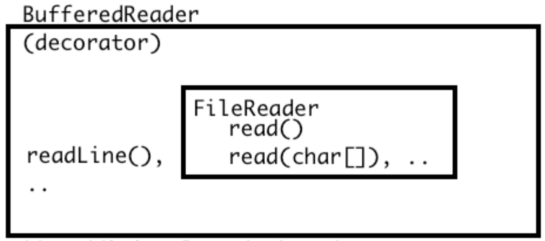
converted to byte -> -1

0000 0000 0000 0000 0000 0000 1111 1111

converted to an int -> 255

java.io.Reader
(abstract class)

- FileReader
- BufferedReader
- InputStreamReader (converts an InputStream into a Reader)

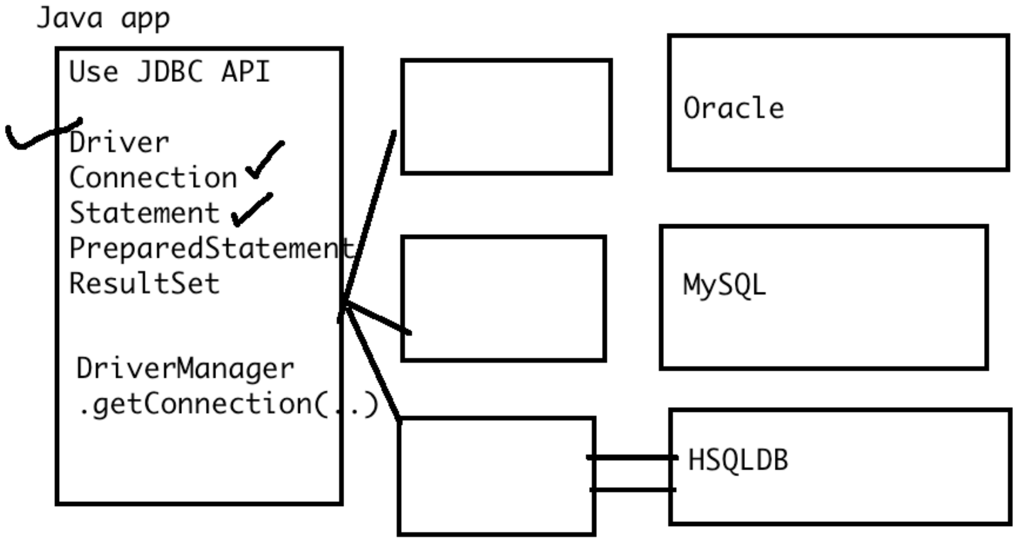


BufferedReader
(decorator)

readLine(),
..

FileReader
read()
read(char[]), ..

adds additional methods, that operate
on the methods of the object it decorates.

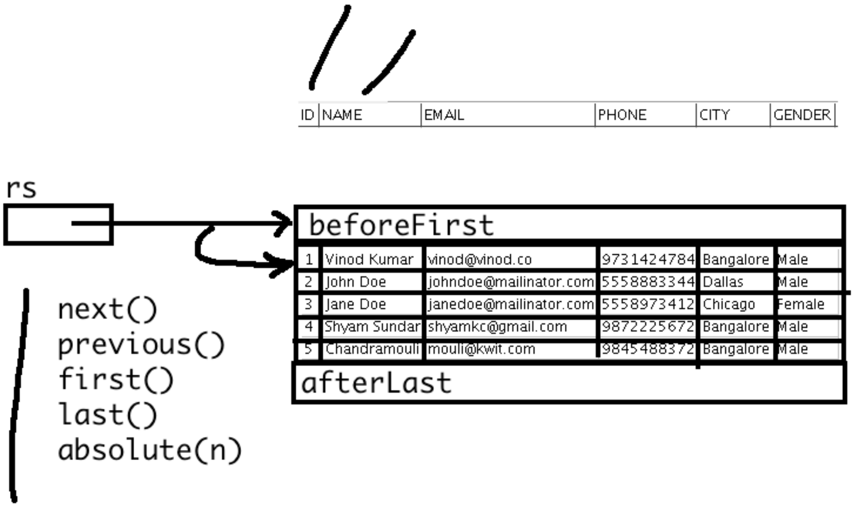


Java app

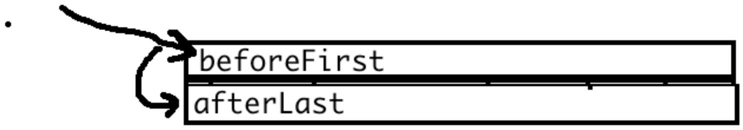
Use JDBC API

✓ Driver
Connection ✓
Statement ✓
PreparedStatement
ResultSet

DriverManager
.getConnection(..)

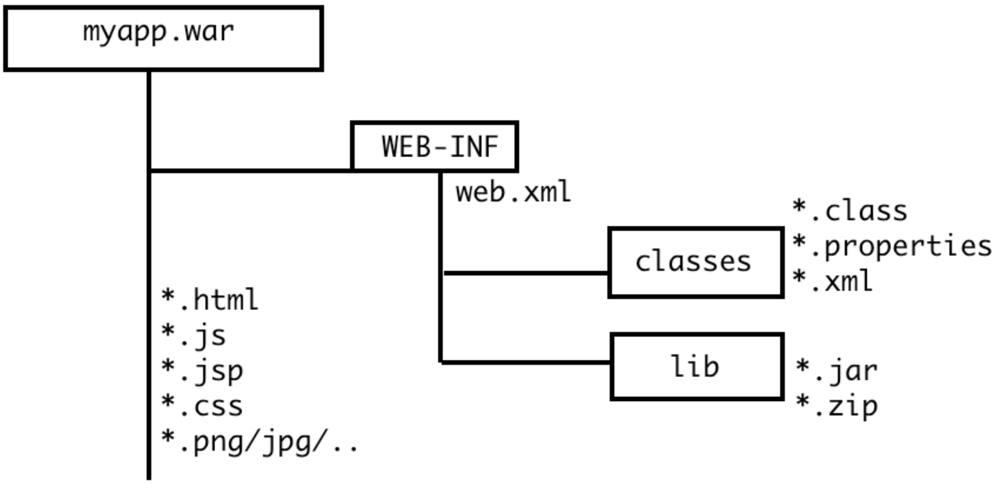


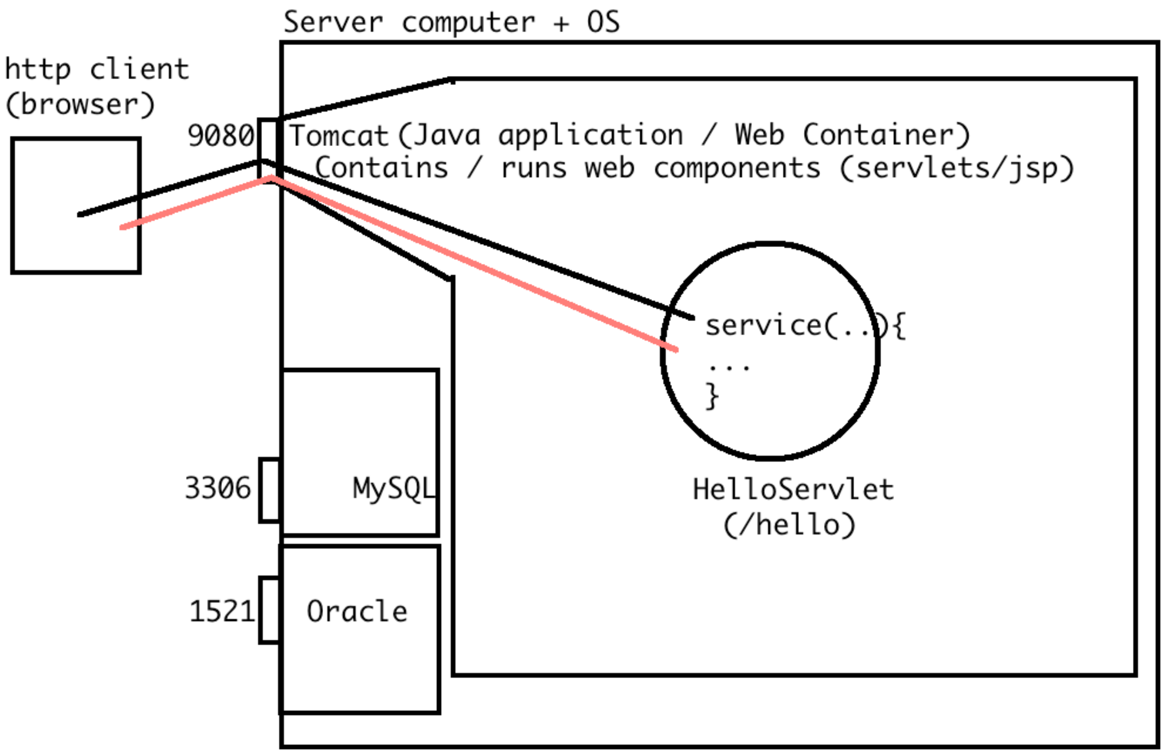
ID	NAME	EMAIL	PHONE	CITY	GENDER
1	Vinod Kumar	vinod@vinod.co	9731424784	Bangalore	Male
2	John Doe	johndoe@mailinator.com	5558883344	Dallas	Male
3	Jane Doe	janedoe@mailinator.com	5558973412	Chicago	Female
4	Shyam Sundar	shyamkc@gmail.com	9872225672	Bangalore	Male
5	Chandramouli	mouli@kwit.com	9845488372	Bangalore	Male

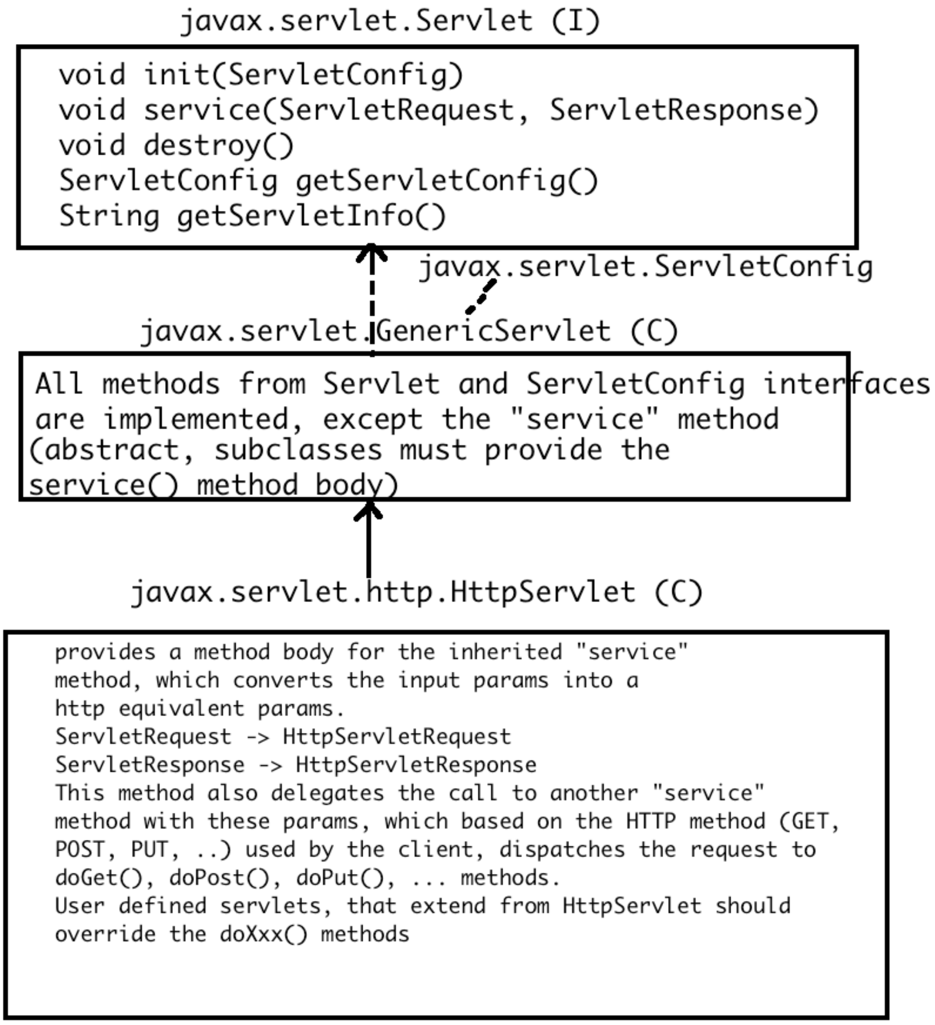


JEE Standard for web applications

WAR (Web Archive)




Server computer + OS
http client
(browser)
9080
Tomcat (Java application / Web Container)
Contains / runs web components (servlets/jsp)
service(..){
...
}
HelloServlet
(/hello)
3306
MySQL
1521
Oracle



javax.servlet.Servlet (I)

```
void init(ServletConfig)
void service(ServletRequest, ServletResponse)
void destroy()
ServletConfig getServletConfig()
String getServletInfo()
```

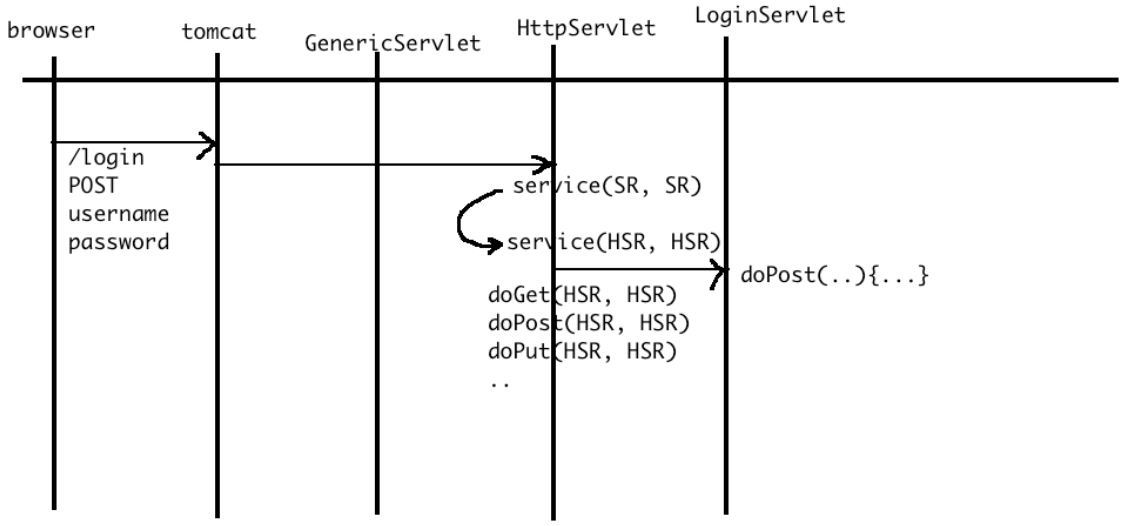
javax.servlet.ServletConfig

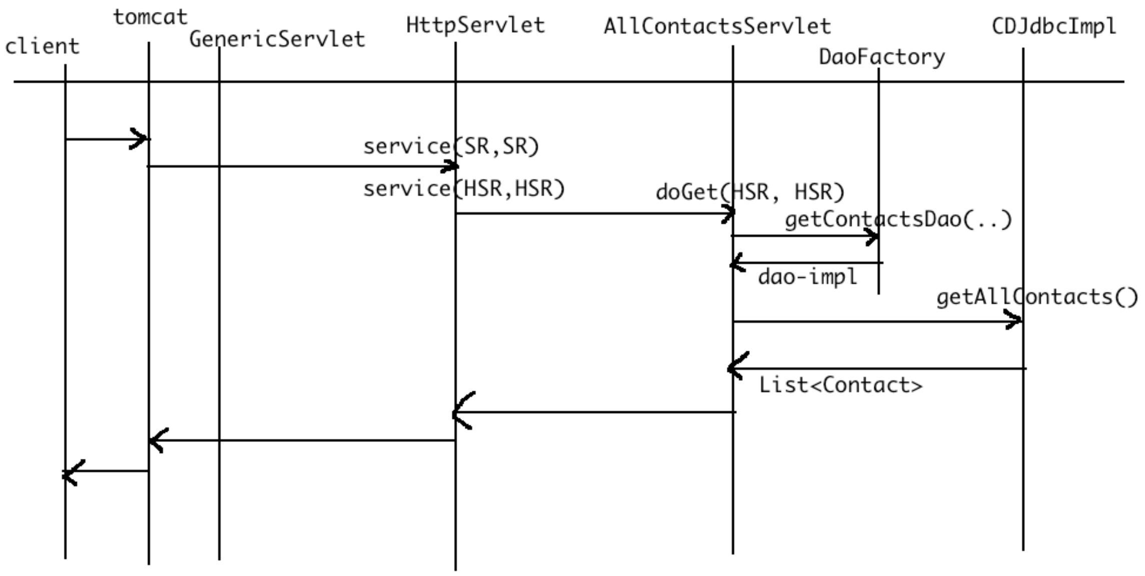
javax.servlet.GenericServlet (C)

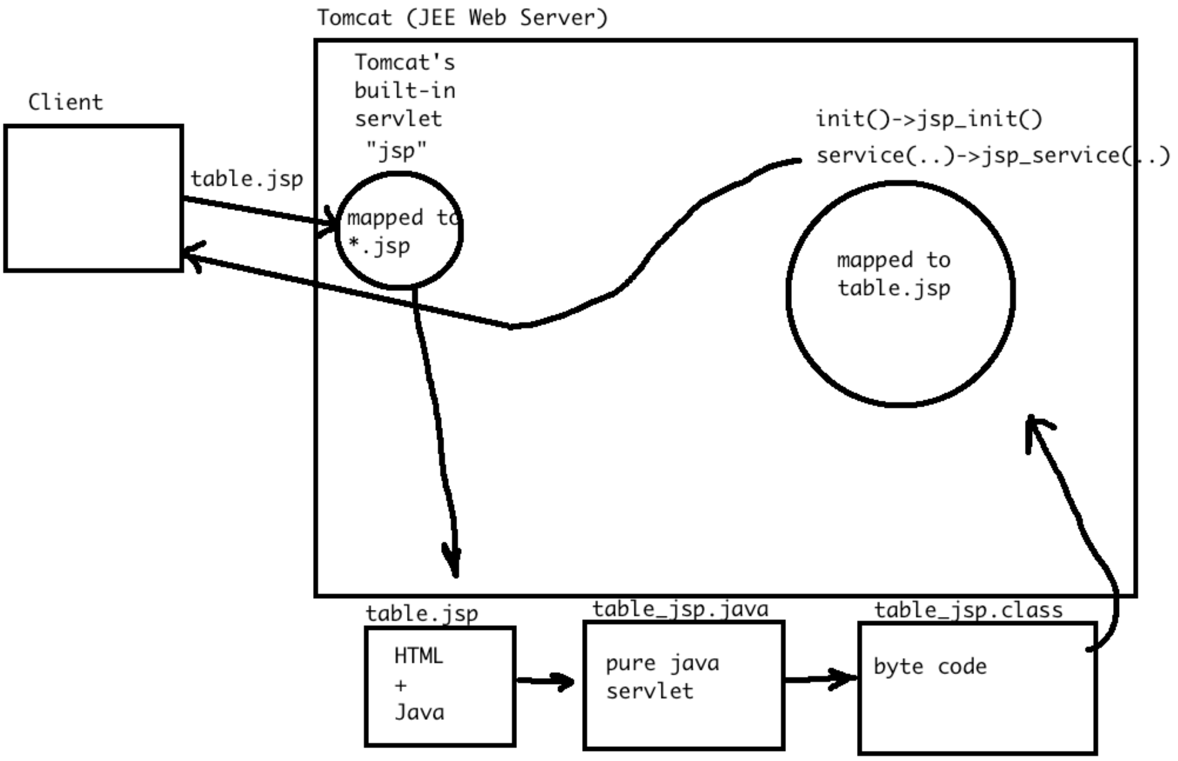
All methods from Servlet and ServletConfig interfaces are implemented, except the "service" method (abstract, subclasses must provide the service() method body)

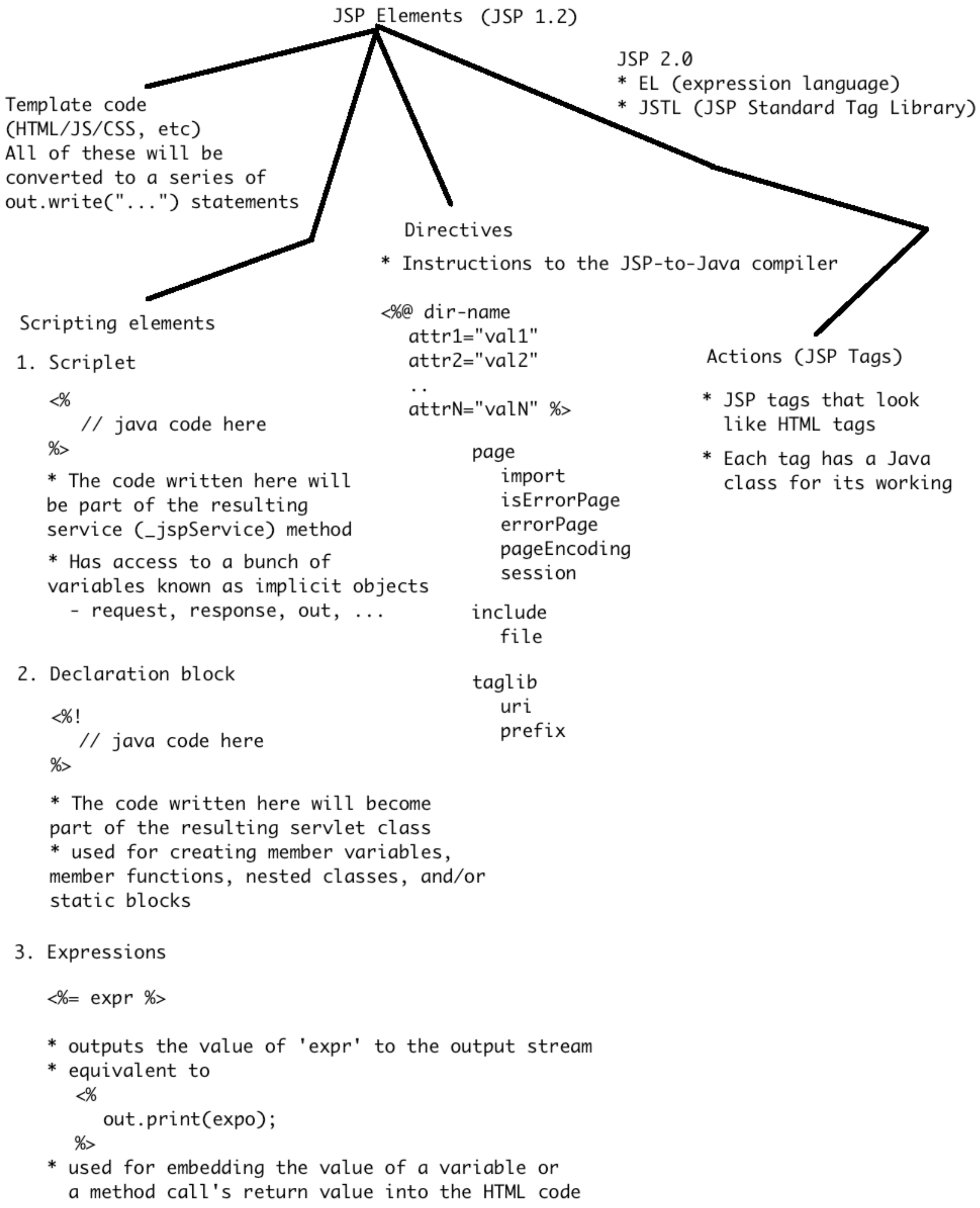
javax.servlet.http.HttpServlet (C)

provides a method body for the inherited "service" method, which converts the input params into a http equivalent params.
ServletRequest -> HttpServletRequest
ServletResponse -> HttpServletResponse
This method also delegates the call to another "service" method with these params, which based on the HTTP method (GET, POST, PUT, ..) used by the client, dispatches the request to doGet(), doPost(), doPut(), ... methods.
User defined servlets, that extend from HttpServlet should override the doXxx() methods


browser
tomcat
GenericServlet
HttpServlet
LoginServlet
/login
POST
username
password
service(SR, SR)
service(HSR, HSR)
doPost(..){...}
doGet(HSR, HSR)
doPost(HSR, HSR)
doPut(HSR, HSR)
..


tomcat
client
GenericServlet
HttpServlet
AllContactsServlet
DaoFactory
CDJdbcImpl
service(SR,SR)
service(HSR,HSR)
doGet(HSR, HSR)
getContactsDao(..)
dao-impl
getAllContacts()
List<Contact>


Tomcat (JEE Web Server)
Client
table.jsp
Tomcat's
built-in
servlet
"jsp"
mapped to
*.jsp
init()->jsp_init()
service(..)->jsp_service(..)
mapped to
table.jsp
table.jsp
HTML
+
Java
table_jsp.java
pure java
servlet
table_jsp.class
byte code



JSP Elements (JSP 1.2)

JSP 2.0
* EL (expression language)
* JSTL (JSP Standard Tag Library)

Template code
(HTML/JS/CSS, etc)
All of these will be
converted to a series of
out.write("...") statements

Scripting elements

1. Scriplet

```
<%
   // java code here
%>
```

* The code written here will
be part of the resulting
service (_jspService) method

* Has access to a bunch of
variables known as implicit objects
  - request, response, out, ...

2. Declaration block

```
<%!
   // java code here
%>
```

* The code written here will become
part of the resulting servlet class
* used for creating member variables,
member functions, nested classes, and/or
static blocks

3. Expressions

```
<%= expr %>
```

* outputs the value of 'expr' to the output stream
* equivalent to

```
   <%
      out.print(expo);
   %>
```

* used for embedding the value of a variable or
  a method call's return value into the HTML code

Directives

* Instructions to the JSP-to-Java compiler

```
<%@ dir-name
   attr1="val1"
   attr2="val2"
   ..
   attrN="valN" %>
```

page
   import
   isErrorPage
   errorPage
   pageEncoding
   session

include
   file

taglib
   uri
   prefix

Actions (JSP Tags)

* JSP tags that look
  like HTML tags

* Each tag has a Java
  class for its working