Udemy Java

Object class

JNI (Java Native Interface)

native keyword

- hashCode() - hashcode is Integer value of each object in java
- equals(Object obj)
- public boolean equals (Object obj) {

    return (this == obj);

}

clone()

toString()

When garbage collector removes object from memory, finalize() method is invoked.

native methods - methods implemented in C, C++.

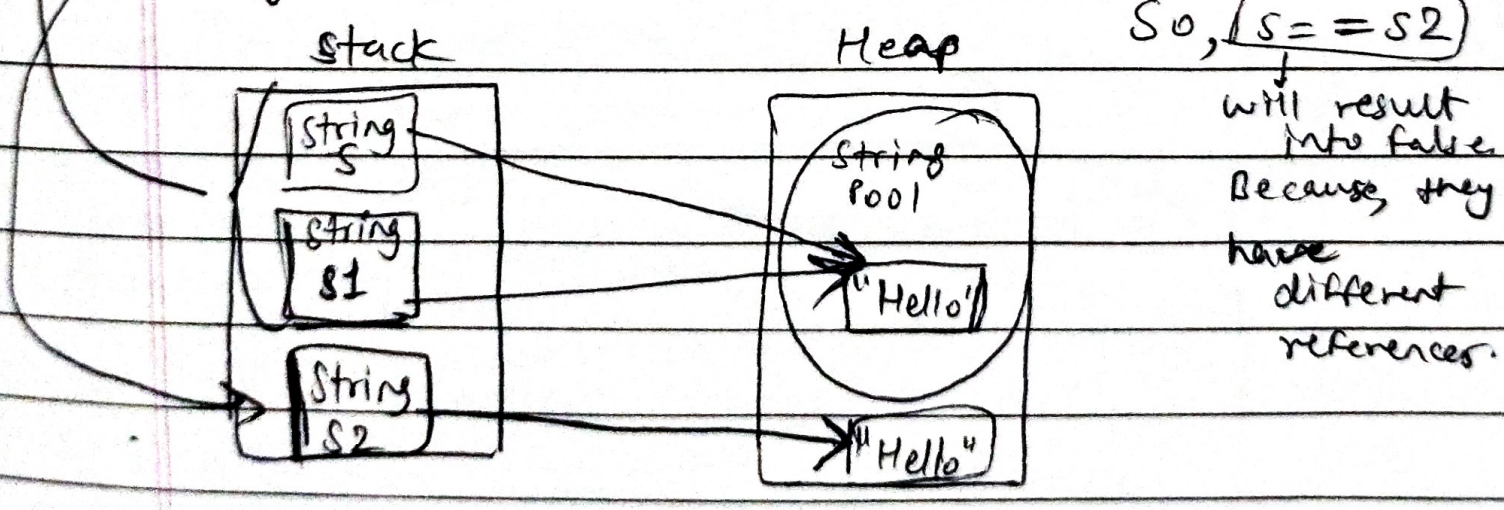
Java udemy

**

String s = "hello";
String s1 = "hello"; } have same reference in pool of strings

But, String s2 = new String("hello");

equalsIgnoreCase



So, s==s2 will result into false Because, they have different references.

Java udemy

DATE / /

```
String template = "Hello, dear %s";
String name = "Swarali";
String formatted = String.format(template, name);
```

SOLID principle for OOP language

S - Single responsibility principle

O - Open/closed principle

L - Liskov substitution principle

I - Interface segregation "

D - Dependency Inversion "

~~Interfaces~~ Collection framework

List, Set, Queue

- List - ordered Collection of elements

↑ Implementations

ArrayList, LinkedList, Stack, Vector

* Set - Collection of distinct* objects

↑ implements

Enumset, Hashset, LinkedHashSet, Treeset

Queue - FIFO principle

↑ implementations

ArrayDeque, LinkedList, PriorityQueue

- List implementations may contain duplicates.

Stack extends Vector.

Vector - thread safe container, it's synchronized,

2 threads can't access it at the same time

CopyOnWriteArrayList

Lambda expression (Youtube - Learn Code With Durgesh)

Lambda expression - Anonymous function
- no name, modifier & return type

```
() ->
{
  // function body
}
```

- reduces the lines of code
- This function can be passed as argument to other function.
- to effectively call APIs

```
(int a, int b) ->
{
  return (a+b);
}
```

```
(a, b) ->
  return (a+b);
```

Java compiler can identify the type of variable passed in arguments. Hence, type is optional.

~~To call lamb~~

Lambda expression is used to implement functional interface in simple & short manner.

Functional interface has only one abstract method.

(Watch the video from 20:20)

```
interface sum
{
public int getSum(int a, int b);
}
```

```
sum s1 = (a,b) -> a+b;
```

```
interface StringLength
{
public int getLength(String str);
}
```

```
StringLength l1 =
    s -> s.length();
```

one argument

length() for string

Runnable interface is functional interface.

→ `public abstract void run();`

It has only one method.

```
Runnable thread1 = () -> {
    for(int i=1; i<=10; i++){
Thread.sleep(1000);System.out.println
    // Body of code
};
```

```
Thread t=
    new Thread(thread1);
t.setName("...");
t.start();
```

Watched till 48:16

Java - Stream API

PAGE NO.
DATE / /

Rest API
Java
Digitalize Framework
Parveen
Hobbies

Stream API (Youtube - Learn Code With Durgesh)

① Stream API - Introduced in Java 1.8.
So, not supported in previous versions

② related to collection frameworks (Group of objects)

different from iostream

iostream - sequence of data, used for read & write

stream api - to process A group of objects

③ To perform bulk operations, reduces code length

① `List<Integer> list1 = List.of(2, 4, 9, 13, 40);`

Immutable list will be created

② `Stream<Integer> stream = list1.stream();`

`stream.filter(`

called as predicate → function with boolean output as true is needed for filtering

③ `List<Integer> newlist = stream.filter(i -> i%2==0).collect(Collectors.toList());`

Combining steps ② and ③ →

OR `List<Integer> newList = list1.stream().filter(i -> i%2==0).collect(Collectors.toList());`

Stream is an Interface

watched till 31:57

A new stream must be generated to revisit the elements of the source.

Pavatpart Stream - filter, collect, print, convert from one data structure to other.

① `Stream<Object> emptystream = Stream.empty();`

`// emptystream.forEach(e -> {System.out.println(e);});`

② `String names[] = {"ABC", "XYZ"};`

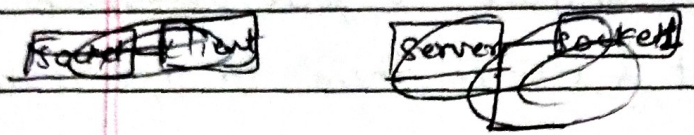
`Stream<String> stream1 = Stream.of(names);`

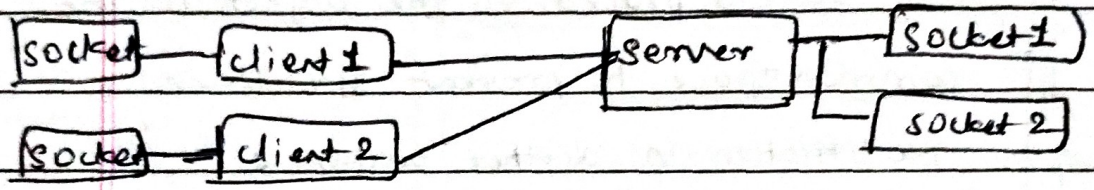
`stream1.forEach(e -> {System.out.println(e)});`

```
import java.net.HttpURLConnection;
import java.net.URL;
import java.sql.Timestamp;
import org.json.JSONObject;
```

java.net package - java networking
- supports TCP and UDP protocols

- **Socket programming in java** (Youtube - Telusko)





Server needed a separate socket for each client
for n clients → n sockets on server.

1. client sends request to server socket & socket is created inside the server.
2. socket should know the IP address of server & port number.
3. After connection is established, send data with the help of Java

① long L2 = 2_000_000_000;

↑ underscores to increase readability

② float f1 = 5.89f;

③ long l2 = 20000000000; // error

long l2 = 20000000000L; ← tell the compiler that it's long.