JAVA AND ENVIRONMENT

1) What is Java?

1-OO Programming Language.
2- JDK toolset.

JAVA Versions?

Java 8 =currently(03/18/**2014** - present).
Java 7= (**2011**-07-28 - 03/18/**2014**)
Java 6 =(**2006**-12-23 - **2011**-07-28)

2) Pure OO??

No, Primitives, Functional programming make it non-pure.

3) JDK vs JRE vs JVM?

JDK—>Java Development kit—> develop java code -> JRE and JVM
JRE -> Java runtime environment -> Client machines to **run** the java code.
It cannot be used to write java code.
JVM -> Java virtual machine-> it is **compile**r. compilation happens at this level
It takes Byte code and runs the instructions. Byte code needs to be tweaked for CPU to run.

How to find version? Command line **—>java —version** =JDK 1.8.0_144

Why are you using Java What is benefit ? WHY java ???

—For example Payton has so many library support and it very easy to use and learn.
—Java is used in many projects as a Backend ,API, etc development and as you environment already set up for java.
—Automation as well selenium+Java combination is popular there is many example
—Java +Selenium is popular and lots of community support

4)Java is platform independent-> Write once-run everywhere.

In Mac- I have JDK -> write .java code -> javac -> .class file ,ByteCode-> JRE+JVM in Any machine can run it.

5-What is java Execution Process:

JAVA CODE --> COMPILE --> EXECUTE

6-What is minimum requirement to write a runnable java code?

--> Class
--> main method
--> save file as .java
java program.

7-What is command to compile:

javac javaProgramName.java
--> it will generate byte code
--> into .class file

JAVA & OPERATORS

8-How to EXECUTE ?

java javaProgramName

9- Class-path/Class-loader -

—> When a program is executed, JVM needs to load all required classes, JVM uses Class loader

—> Looks for Classes in class-path.

10-Java is "STRONGLY TYPED LANGUAGE" ?

--> YES you need to pre-define the data type of a variable and declare before you can use it.

11-Are JAVASCRIPT and JAVA the same?

They are both similar and quite different depending on how you look at them.

—-Java is an Object Oriented Programming (OOP) language—>run in a VM or browser -java code needs to be compile

—-JavaScript is a scripting language used design website and run on a browser— scripting languages are all in text

12-How do you DECLARE in Java?

Declaration==>String str;

Assignment==>String str="name";

==========

SIMPLE ASSIGNMENT OPERATOR

= Simple assignment operator

ARITHMETIC OPERATORS

+ **Additive** operator(also used for String concatenation
- **Subtraction** operator
* **Multiplication** operator
/ **Division** operator
% **Remainder** operator. MODUELE

UNARY OPERATORS

+ Unary **plus** operator; indicates positive value
— Unary **minus** operator; negates an expression
++ **Increment** operator; increments a value by 1
-- **Decrement** operator; decrements a value by 1
! Logical complement operator opposite the value of a boolean

EQUALITY AND RELATIONAL OPERATORS

== Equal to
!= Not equal to
> Greater than
>= Greater than or equal to
< Less than
<= Less than or equal to

CONDITIONAL OPERATORS

&& Conditional-AND TT—>T
|| Conditional-OR. FF—>F

Exclamation mark ! Not Opposite

MAIN METHOD. public static void main(String[] args)

1-What is main method?

-> Entry point to the execution.

2-Why MAIN?

> You can compile a class without main method but you cannot run it.

-> Jdk/Jre looks for it when executing the code.

3-Why main method is STATIC?

—> is it Static -> it can be called directly.
—> It must belong to the class in order to run the class.
—-> Main method can hold static and non-static.
Because main method is called by JVM before any object are created. Therefore it has to be static in order to invoke main method.

4-Can you have MULTIPLE main methods?

In Java, you can have just one public static void main(String[] args) per class. Which mean, if your program has multiple classes, each class can have public static void main(String[] args) . See JLS for details.
A class can define multiple methods with the name main

5-Can main method be PROTECTED?

Yes, but it can not be taken as entry point of your application. ... But it will not run as entry point of the program. Java looks for the public main method signature. If any of the modifiers is different, then it will assume it as some other method.

6-Can you CALL a main method from a different class?

Yes-only if main public we can call it

7-Can you OVERRIDE a main method?

No we can not, main method is static and cannot be overridden.

---

public static void main(String... args[]){. }

- public – it is the access specifier.
- static it allows main() to be called without instantiating its belong the class
- void – it affirns the compiler that no value is returned by main().
- main() – this method is called at the beginning of a Java program.
- String args[ ] – args parameter is an instance array of class String

CLASS- OBJECT

What is a different CLASS vs OBJECT ???

Class: Is a blue print like template .
More general, Its store data/variables and Behaviors/Object.
Help the reuse the code
We can keep create the object.

Exm:
```
public class Student{
    String name;
    public void study(){}
```

Object: is an instance of the class.Object comes from the class/create from a class/

Exm: Scanner **scan** = new Scanner(system.in); in **HEAP** memory.

scan is a reference to the new scanner object.'**scan**' has all the power of Scanner class. Because it is object from Scan- new Class.

==========

STRING

Mutable means: Once create can be modified/changeable in memory in heap pool
Immutable -> unchangeable in memory/Can not modified

1-Strings are mutable or immutable? ->String is IMMUTABLE/UNCHANGE

```
String str="abc";   //unchangeable just create new one memory:(
str=str+"d";
str=str.toUpperCase();
```

1-abc-(unchanged still there) 2-d " 3-abcd " The last version is 4-ABCD

NOT: If we are going to create more String for the memory management we use . They change by updating object String str="ab"; String str="cd; ~~"ab"~~

String BUILDER: it is new -fast - no synchronized
String BUFFER: is old-Thread Safe—synchronized-

2- String Builder, String Buffer. (Those are classes) =>MUTABLE/CHANGBL

```
StringBuilder sb=new StringBuilder("abc");
sb.append("def");
String str=sb.toString();
str.equals(sb);
```

NOT:StringBuilder is mutable so whenever there is lots of string manipulation , this is better choice.

Code --> Running in 10 instances/threads at same time. When an object is

Thread-Safe - it means that particular instance will not be disturbed by other threads. And it guarantees independent execution.

Code/Objects --> Thread1. Like --> Thread2 —> --> Thread3 ——>

PRIMITIVES 1

1) Java is strongly types language:

Whenever you want to use a variable -
you need to predefine the datatype and declare it.
int num;
boolean b;
in **BB Script** ==> dim num=you can write all type of variable,
but in java you should predefine like string int Boolean ..

2) Primitives vs non-primitives?

Primitives-> only data/value, no actions/ no behavior
Where in memory? stack
Objects->Can hold data, process the data, perform some actions etc.

3) Primitives: How many types ? 8 types

Integers numbers: byte, short, int, long
Floating point: float ,double
Boolean: True False
Character: char. 'a' also can have "45543" and see the by one buy

4) primitives to WRAPPER classes?

int > Integer
short > Short
double > Double

```
int i=10;
Integer n=i;
Integer num=200;
```

boolean—>Boolean.

Character,Long

```
Integer i = Integer.valueOf("123");
Byte b=new Byte(12);
```

5-What is casting?

is taking an Object of one particular type and "turning it into" another Object type.

```
int n=100;
double d=n;          //implicit casting

short sh=(short)d; //explicit casting
byte > short > int > long
short sh2=35;
int n2=sh2;
byte b=(byte)n2;
```

PRIMITIVES

6) Auto-boxing/unboxing (Wrapper classes)

Auto-boxing -> primitive— to—> object taking primitive value and assigning to Wrapper class obj

```
Integer intVal=44; //new Integer(44);
```

Auto-boxing --> is a process when you take a prmtv value and assign into wrapper class obj

```
int i=10;
Integer n=i;
Integer num=200;
Integer num2=new Integer(400);//NO BOXING
```

Unboxing -> Object-to->primitive taking wrapper class instance and assigning to primitive type

```
int i=intVal;
```

Un-boxing -> is a process when you take Wrapper class object and convert to primitive.

```
Integer num2=new Integer(400);
Integer num=200;
 int i=num2;
```

7-CASTING types.

Implicit casting: kendiliginden olan normal kuralla

```
short j=343;
 int i=j;
byte>short>int>long>float>double
```

Explicit casting:

```
short sh=550;
byte bt=(byte)sh;
byte>short>int>long
```

What is casting? is taking an Object of one particular type and "turning it into" another Object type.

8-How do you swap values? Write notes ……

Java passes stuff by **value**, which means the variable your function gets passed is a copy of the original, and any changes you make to the copy won't affect the original.

Exm: void **swap**(int a, int b) { int temp = a; a = b; b = temp;

// a and b are copies of the original **values**.

Swaping Value exm

```
int a=10;
int b=5;
a= a+b; 10+5——>15
b= a-b; 15-5——>10
a= a-b;  15-10—->5.          a=5.     b=10
```

STRING & MANUPULATION

```
String str = "This is the test string";

Sys.out(str.length()).     //gives the size
Sys.out(str.charAt(3));//returns char of given index-Opzt indexOf

Sys.out(str.concat("  ---sonuna ek--- "));//merges to string
Sys.out(str.contains("t"));   //is it contain t inside
Sys.out(str.startsWith("This"));   //CASE SENSITIVE\
Sys.out(str.endsWith("ing"));    //true

      String str1 = "Hello";
      String str2 = "Hello"
      String str3 = "Welcome";

                  // compares values of strings ignoring the case
Sys.out(str1.equalsIgnoreCase(str2));
Sys.out(str1.equals(str2)); //HELLO HELLO --->true
Sys.out(str.indexOf("h")); //returns index of given char //1
Sys.out(str.isEmpty());       //false

 String str5= "       Here is a empty space";
 Sys.out(str5.trim()); Cuts off empty spaces on both sides
                    //  Here is a empty space

 Sys.out(str.toUpperCase());
 Sys.out(str.toLowerCase());

 Sys.out(str.replace("t", "k"));Replaces with givencharString
                                    //All t, replies with k

 Sys.out(str.substring(2)); Cuts the string, returns string
                          starts zero // "his is the test string"
 Sys.out(str.substring(5, 10)); //cuts between index
   IndexOutOfBoundsException
  -If the beginIndex is less than zero OR beginIndex > endIndex OR
endIndex
   is greater than the length of String.

char[]charArray=str.toCharArray();it gonna take all chrcter & give it Array.
        for(int i=0; i<charArray.length; i++) {
         Sys.out("This is  : " +  i  +   charArray[i] );
 //This is  : 0T-    //This is  : 1h-    //This is  : 2i-       //This is  : 3s
```

STRING METHODS

1-String Class: comes Java.lang.String.import automatically thats way we don't import strings libraries

2-How to create String object?

-String str="abc"; Store in String **pool** in **Heap.**

If "abc" already exists, it will reuse, if not it will create in String pool

-String str=new String("abc");

3-String ———>TO ———> int?

- **valueOf** this method converts different datatypes into String.

```
Integer x = Integer.valueOf(str);
```

-**parseInt** returns primitive int

```
int y = Integer.parseInt(str);
```

4- int ——> TO———>String ?

-String.valueOf: The method converts int to String.

```
int i=10;
String s=String.valueOf(i); //Now it will return "10"
```

-Integer.toString() method converts int to String

```
int i=10;
String s=Integer.toString(i); //Now it will return "10"
```

5-Convert String Array——> to——> String ?

```
String[] testArray = {"Apple", "Banana", "Mango"};
String testString = Arrays.toString(testArray);
System.out.println(testString);//[Apple,Banana,Mango]
```

6-Convert Array List To String comma separeted ??

```
List<String> cities = Arrays.asList("Milan", "London", "New York" );
String citiesCommaSeparated = String.join(",", cities);
Sys.out(citiesCommaSeparated);
//Output: Milan,London,New York,
```

7-How to reverse String?

```
String s = "Sevil";
for(int i = s.length() -1; i >= 0; i—) {
System.out.print(s.charAt(i)); }          // liveS
```

8-Splitting the string and printing it as a list ??

```
String s = "Cybertek";
String [] arr = s.split("b");
System.out.println(Arrays.asList(arr));
s.indexOf('e',s.indexOf('e')+1);
```

CONTROL STATEMENTS —> CONDITIONS

How many different ways to check conditions in java?

1- If ConditionsStatement: if-else if-else if -else-**Logical operators**: && || !

2- Switch Statement: Switches from one condition to another

Support: switch(num) Strings, Enums, int, chars,,short, byte

Test **equality**.We use switch If there is a multiple things to check is it equal or not .Default is not mandatory. And it can be anywhere in switch block.

3- Operator: Condition ? true: false; boolean b=false; String str= b? "yes": "no";

IF Vs Switch

-If condition check by one by , directly

-Can test for any kind of comparison.

-Switch go and find

-Read easy and fast :)

-Only check == condition

SWITCH STATEMENT

```
public class Test {.  main method
      // char grade = args[0].charAt(0);
 char grade = 'C';
 switch(grade) {
  case 'A' :
     Sys.out("Excellent!");
     break;
   case 'B' :
   case 'C' :
        Sys.out("Well done");
==========================
        break;
  case 'D' :
        Sys.out("You passed");
 case 'F' :
        Sys.out("Better try again");
        break;
   default :
        Sys.out("Invalid grade");}
  Sys.out("Your grade is " + grade){}
```

IF CONDITION

```
int a = 10;
int b = 5;
if (a > b){
     Sys.out.("a is greater than b");
else{
     Sys.out.("b is greater than a");
Output is // a is greater than b
```

Continue vs. Break

Continue→Skips current iteration

Break →Exits the loops/ breaks the loop

CONTROL STATEMENTS —->LOOPS

LOOPS: While , do while, for , foreach (enhanced for loop)

While→ Runs according to condition

Do while → Runs at least once

For → When know how many times we are going to loop,can handle collections except fews Map<String,String>
For loop. Can backwards as well 10 to 1

Foreach→ Loops through all elements of collection we passed.
Support only works forward iteration. 1—-to —10

Iterator vs foreach

** iterator can modify original collection. — whereas foreach cannot.

For vs for Each?

-Used when we know how many time —Only use with some sort of collection.

-Can also handle collections. & it ll iterate each item one by one
Map<String,String>

—Can loop backwards as well. —-is only forward iteration

1-How would you print odd numbers from array ?

```
int[] nums={1,2,3,45,5,6,784,523,34};
for( int num : nums ){
  if(num % 2 > 0){
  print num;  }.  }
```

2-How do you end the loop, before condition is met? By **Break;**

```
int[] nums={2,3,45,5,6,784,523,34};
for( int num : nums ){
    if(num % 2 > 0){
    print num;
break;
```

Continue: Continue to next iteration of loop.

```
for( int num : nums ){
if(num % 2 == 0){
   continue;}
```

MODIFIERS

1-How many acres modifier?? 4

private >default > protected > public >

Not: instance and class(static) variables, non-abstract methods, abstract methods can have public . protected and default

All 4 access modifiers - instance variable, methods

Can't apply to local variables

2-Non-Acces Modifiers?

static- final- abstract -syncronized,

Not: Local variables cannot have access modifiers

Type	in class	In Package	Between Package Sub Class	World
Private	X			
Default.	X	X		
Protected.	X	X	X	
Public.	X	X	X	X

==========

FINAL VS FINALLY VS FINALIZE

FINAL -> When applied to something it means value can not be changed

if its final class: cannot be extended / inherited

final variable: value cannot be changed. When declaring final variable you must initialize it.

final method: cannot be overrides. Implementation can't be changed.

FINALLY BLOCK -> it is inside the exception handling.

-Finally block will be executed no matter what happens in try/ catch block.

-Exception handled or not handled finally() block will be executed.

-Only way not to run finally() block is

System.exit(1) This statement terminates the program.

Why Usefull: I use because it allows the programmer to avoid having cleanup code accidentally bypassed by a return, continue, or break.

*****It **must** be with associate with try block- it is **optional** for try-catch block

STATIC. Shared members.

1- What Static keyword/Members? Belongs to class directly and can be called with class name, without creating an object. Does not ***depend*** on object

-Static can only ***refer*** to static members. Non-static can refer to both static

-A class can have ***multiple*** Static blocks, which will execute in the same sequence in which they have been written into the program.

-They can be ***called*** through classname without creating object.

-Objects of a Class also can ***call*** Static members

-Local variables ***cannot*** be static

TIPS : In test automation, we will store our helper methods in one class and mark them as static. It is a idea to keep them under utilities package.

2-What can be static?

-static variables/Class variable: Can be accesses without an object by using the classname

-static methods: Static methods can be called using a classname and an object

ex: ClassName.StaticMethodName();

-static blocks: Code in static block runs only once whenever you try to use the class. Its run before everything else. before any constructor.

-static inner classes (java 8)

==========

FINAL VS FINALLY VS FINALIZE

-FINALIZE() GC (Garbage Collector)

-In java, we don't have to worry about cleaning up the object from memory.

-JVM automatically cleans up the created object. Whenever object does not point to any reference anymore then JVM calls finalize() method to delete that object to clean up memory.

- method is defined in java.lang.Object

- **!!!** If you are overriding finalize method then it's your responsibility to call finalize() method of the superclass,

-if you forgot to call then finalize of super class will never be called.

OOP OVERLOADING -OVERRIDING

1- Method Overloading: 2 method with Same method name
different parameters, signature must be different

Also call **Ad-Hock** polymorphism
Return type can be different (void int boolean string)
Access modifier can be different

Q- Is return type part of method signature?
No return type is not part of it. That's why we cannot overload the method just changing the return type.

Q-Example Selenium Method??
elementToBeClickable**(By locator) (WebElement element)**
wait(), wait(timeout), wait(timout,nanos)

2-**Method overriding:**
-**Method signature** must be same, name+paramaters
-**Return type** must be same, only narrower types are allowed
-**Access modifiers** -> can be same or more accessible
-**Static** methods cannot be overridden - they can be hidden. and depending on where it is called, it will call that version
-**Variables** cannot be overridden - they can be hidden.

Overriding --> in SUB Class

When a class inherits a method from a super class and changes method
implementation by keeping same signature and return type.

1. Happens between sub and super classes. inheritance
2. Method signature must be same
3. if return type is primitive, must be same. if return type is object, it must
be same or sub type
4. Access modifier cannot become more restrictive
5. Override: ***throws Exception*** is not allowed.
-If a method in Super class throws exception, you cannot make that exception type broader.
6. **final** method can not override
7. **Static** methods cannot be overridden. They can be hidden. and depending on where it is called, it will call that version.
8. **Constructor** methods cannot be overrated.
9. **Only methods that are successfully inherited can be overridden**

CONSTRUCTOR (yapilandiricilar)

Constructer-- Special kind of method, used to create and object.

-Every time we use **new** keyword constructor is called.

-It has the **same name** as the class

- It should **not return** a value not even ***void***

- Every class will have **default** constructor by JVM.(No argument cnst

-once constructor created manually default one is not there anymore

Q-Why can't this() and super() both be used together in a constructor?

-this(…) will call another created constructor in the same class

-super() will call a super constructor . If there is no super() in a constructor the compiler will add one implicitly.

Q-Constructor chaining?Consider a scenario where a base class is extended by a child .Whenever an object of the child class is created the constructor of the parent class is invoked first.This is called

-Constructer always call the base class contractor

Q- Why When do we invoke? learn

if super class could have private fields which need to be initialized. by its constructor.

==========

ABSTRACTION: Defining only behavior but without implementation.we hide datails

We don't need to see the implementation just we need to see functionality.

public abstract void execute();

We don't need to see the implementation just we need to see functionality.

I know there is some method available to use, I don't need and care details.

For the reading the this method there is 2 way

1) ABSTRACT classes: Start to key word abstract.When we put abstract key in class name it will be abstract class exam: **public abstract class AbstractCars {}**

2) INTERFACES—>collection of abstract method

Implement the interface they are actually signing a contract with underpays that I am going to implement all those methods that you define in yourself.java not support multiple inheritance and INTERFACE comes because by using multiple interface one class can implement all those interfaces , and can define all the methods of relabeling multiple interfaces

INTERFACE can not implement methods!!!!!

Interface can also contain the variable the constant and all. (We will cover java 7) (no default and static method)

OOP ENCAPTULATION — — — INHERITANCE

ENCAPSULATION: Hiding the data.Ths data will be hidden from the other classes. Can be accessed with getters and setters.
private variable declaration

INHERITANCE => using some other classes members.
=> **Why we use ?** Reusing ready methods/ behaviors / actions.
=> If you don't like the behavior then we are able to provide with our own implementation as well.
=>Using '**extends**' keyword we can inherit other classes members.
=>Can we inherit more than one —> yes
=>**public-protected** all can inherit,
=>**default** only can inherit if sub&super class are in the same package
=>**private** members are NEVER inherited need to import if different package
=>Final class- Contractor con not be inherited

Q-**Can a super class referenced variable hold an object of sub class?**

```
A var = new C();
var.m1();
```

yes through polymorphism

```
B b = (B)new A(); ClassCastException.
 new C()
```

this() vs this

this()->constructor for current class /kind if overloading constructer
→used to call another constructor from a constructor **in same class**
this → representing the object of current class
→ used for refer to instance variables and methods ***

Super. vs super()

super() will call parent class constructor-Call base class constructor calling the super class's default constructor can be used only inside the constructor.

super is representing the object of superclass
call parent members

OOP ABSTRACTION ———POLIMORFIZIM

ABSTRACT CLASS VS INTERFACE

Abstract class: Can have both abstract and non-abstract methods.
Or can have pure abstract or pure non-abstract methods.

Interface → can have **abstract - default- static** method

java 8: **Default** and **static** methods have method body

→ Both abstract classes and interfaces cannot be instantiated.
Because they might have methods without bodies (implementations)

→ ***Abstract classes*** => we use **extends** keyword →

-->***Interfaces*** => we use **implements** keyword

** Interface can extend to another interface.

** ***Concrete class***(class that is extending to abstract class or implementing interface) must provide with an implementation for abstract methods.

*****A class can only extend to one class NO MULTIPLE INHERIT
but can implement multiple interfaces.

POLYMORPHISM

→ one object having different forms. Cell phone ex/ driver ex

Left side is reference type =Right side is object type

Exam:

Eat h = new HumanEat();

h.eat() → This will execute humanEat implementation of eat() method.

Ex: must give example:

List< String> li = new ArrayList<>();

List → is reference type ArrayList → is object type

*** Object type has to have is-a relationship with reference type.

** **UPCASTING** happens automatically

*** **DOWNCASTING** must be explicitly written.

Static polymorfizm: Compile time—> overloading in same class

Dynamic polymorfizm: Run time **—>** Overriding. in different class