CYBERTEK
Sevim Surucu, 2019

September 28th

Typing
typing.com
ztype.com
nitrotype.com
typeracer.com 30 mins a day

English
Record yourself for speaking English
duolingo.com
lyricstraining.com
breakingnewsenglish.com

Text editor
sublimetext.com download this

Test preparation
quizlet.com. test preparation

Code practicing
Codingbat.com
hackerrank.com do 5 practice everyday

Conceptional knowledge questions
Posting your codes everybody see it you will get feedbacks
edmodo.com
Class code: 2ernsr

Books
Java certification book
Walter Savitch java programming book
Java homebrew

October 1,2019

What is computer?

An electronic device(hardware) for storing and processing data, typically in binary form, according to instructions given to it in a variable program(software).

Algorithm

A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.

Pseudocode

Line by line stating of the steps of algorithm in English

Computer program : list of instructions

Programming languages

1. Low level

Hard to read and understand
such as IBM 360 assembler, PDP-10 assembler, Intel x86 assembler, etc.

2. High level

Most likely to read readable.
C, C++, Java all are high-level languages only.
because it can be read and written easily by humans.

Compiler translate your program to machine code

Compile=> turn this into machine language it does not show the result
Execute=> run it shows the result

Source code => regular written code

IDE- Integrated Development Environment
intelliJ IDEA, NetBeans, Eclipse..

Basic Programming Concepts

- Storing data
- Manipulating data
- Sequence
- Conditional branching
- Repetition

File System

Folders
Subfolders
File extension
For Mac to find path right click "get info"

MAC TERMINAL - COMMAND LINE

pwd shows you where you are (print working directory)
ls list everything from where you are
cd change directory
cd .. go to previous directory/ parent directory
clear clear
mkdir create a folder
rmdir delete a folder

Program Execution in Terminal

1 - compile it
Java always comes with compiler
2- we can execute the .java program on terminal /command line
3- open terminal
4- javac FileName.java + enter.. javac means java compiler
5- after compiling compile, compiler creates FileName.class file
6- for running the class file write java FileName + enter
7- if you don't have any error it means compiled perfectly

8- you can add print statement to see value
9- every time when you change the code recompile with javac FileName.java
10- then run "java FileName" class in terminal. It runs the FileName.class file

JAVA VIRTUAL MACHINE

Java Virtual Machine (JVM): is a compiler between code and computer every computers has JVM for running java programs
Java code works everywhere

October 2nd, 4th, 5th 2019

IntelliJ IDE (Integrated Development Environment) is a software that you can compile your java program.

1. Java project - every related files in the project. You should have maximum 3-4 projects. You will have only one project in job for working on it.
If you want to delete your project from intelliJ, folder will be stay in computer. You can go /user/sevim/ideaproject to see your files

*if you have any problem with other classes your program doesn't work it just compiles all the programs
*compilation error when your code won't compile successfully intelliJ will detect and tell you with red line under code.
*Creating another project on same place is not possible on intelliJ. You need to open the project in a new window.
*refactor - it lets you give/change different name for all of the same variable. Control+T
*All class names should be start with uppercase.
*Every program in java starts with java main method
Project - src - package - java class

2. Primitive Data Types

Primitive data types

Byte	1 byte
Short	2 bytes
Int	4 bytes
Long	8 bytes
Float	4 bytes
Double	8 bytes
Char	2 bytes
Boolean	1 bit

```
char first = 'a' ;
boolean check = false;
float num = 0.5f;  you need to use f at the end
```

Float values have between 6 and 9 **digits** of precision, with most **float** values having at least 7 significant **digits**. Double values have between 15 and 18 **digits** of precision, with most double values having at least 16 significant **digits**.

```
long num = 4L;    you need to use L at he end
```

```
double num = 0.5;
```

double is a 64 bit IEEE 754 **double** precision Floating Point **Number** (1 bit for the sign, 11 bits for the exponent, and 52* bits for the value), i.e. **double** has 15 decimal **digits** of precision.

```
int num = 25;
```

String

String is not primitive data type, it is object that can store bunch of letters and characters .
String keyword start with CAPITAL letter.

String word1 = "hello"
String word2= "world"

System.out.print (word1 + word2); concatenation

3. Type Casting

Sequence smaller data type to bigger data type

1	2	1	2	4	8	4	8
Byte	Char	Byte	Short	Integer	Long	Float	Double

Be Careful Bears Shouldn't Ingest Large Fluffy Dogs????

Implicit casting. Smaller data type to bigger type java change it
Explicit casting. Bigger data type to smaller you cast it with

```
double height = 6.5;
int intheight = (int)height; explicit casting
```

Height will be 6 it rounds down never rounds up

3. Arithmetic Operators

- , + , / , * , %

4. Print statement

System.out.println();

5. Concatination or Operation

```
System.out.println (num1 + num2 + "hello world" + num1 + num2);

// first num1 and num 2 is a operation after string "hello world" is a concatination it is just adding them together. Any data type concatenate with string becomes string.
// output 30 hello world 10 20
```

Concatination : combining strings with "+ "
Ex: System.out. println ("My age is" + age);

6. Assignment and Declaration

Variable - is something that can store a piece of data.
Must declare a variable type and name before it can be used. Only declared type of data can be assigned. When you declare variable it is allocating the memory means creates an empty storage. There is no any data in it yet. You need to name your variables properly. You need to understand that what kind of information it has. ex: int age;

Variable declaration
int a; data type and variable name

Assignment - giving some data into your variable. When assignment happens with
"=". assignment operator

```
int age;
age = 25;
```

int number ; declaration of variable
number = 5;   assignment of variable

int number = 5 ; you can declare and assign variable at the same time

int num1, num2;
string text1, text2;   you can declare multiple variables

Scope of variable declaration : only within the curly braces

* Statements : every line of code ends with semi colon.

7. Short hand assignment operations

```
int a = 5;
a = a + 5;
a += 5;
```

You can write it like a += 5 short hand operator

If you want to increment or decrement the values of numbers by certain amount you can always use shorthand assignment operator.

8. Scanner

Hard coding : when you are coding with static (not changing) values. Fixed constant value that was written by the programmer

Ex:

```
String name = "James";
String lastName = "Bond";
String location = "London , UK";
boolean isHeOk = true;
```

Scanner allows you to read values from outside of the code.

Scanner variableName = new Scanner (System.in);

We will use method : nextLine ();
for getting string from computer

We will use method : nextInt ();
for getting integer from computer

Compilation error: Code can not be compile. This happens before you run the program.

Runtime exception/error : This happens when program is running. The worst thing ever. User inputs something you could not get. Your program will not continue.

When you want to write different data type from console then expected data type it is run time error we need to handle it
* you can google any error message

```
Exception in thread "main" java.util.InputMismatchException
        at java.util.Scanner.throwFor(Scanner.java:864)
        at java.util.Scanner.next(Scanner.java:1485)
        at java.util.Scanner.nextInt(Scanner.java:2117)
        at java.util.Scanner.nextInt(Scanner.java:2076)
        at scanners.Data.main(Data.java:18)  your problem
```

import java.util.* * this called "wild card" it brings every thing from util package

scanner.close(); closing the input from console

9. Incrementing - Decrementing

Increase by 1. age++ //age = age + 1;

++age pre increment first adds 1 then print
age++ post increment first prints then add , adds 1 and work in next line.

```
int age = 10;
age += 6;  // age = age +6; short hand assignment operator
age +=1;    // age = age + 1  same with age++; age++;
```

10. And / Or && ||

True && true = true All of them should be true for true
True && false = false
False && false = false

If you use only & or only | it will going to look either side same thing but execution is different.

True | | true = true. At least one should be true for true
True | | false = true
False | | false = false

11. == equal to

Comparing two values equal to

12. != not equal to

Comparing two values not equal to
for string you can not use "=="

word1.equals(word2);

13. Greater than > - less than <

14. ternary operator ? :
String name = (condition ? if it is true do this : if it is false do this)

String name;
String name1= name.length()>5 ? name : "Bob";
if name length is bigger than 5 give name to name1 if it is false give Bob to name1

IF ELSE STATEMENT

```
if (boolean_condition) {
}
else{
}
```

It decide flow of the application.
if the condition can be used along without "else"
condition must be a statement that returns boolean condition
if you put any code between out side of if else blocks if else will not be compile

else if statement is used when we have few conditions to check before we go to else statement

nested if else statement

SWITCH STATEMENTS

switch statement are used when the flow of program is based on something other than boolean condition

- switch statements will be based on Controlling expression
- controlling expressions can be byte, string , char, integer, enum, constant values
- case label should match with controlling expression data type
- keywords switch _ case _ break
- you can have any number of cases
- you can have any number of statements in the case
- break keyword terminate the switch statement
- you don't need to break, it is optional, if you don't have break it is going to execute the following cases until break!!
- default you don't need to default, it is optional
- you can not write && ||

Syntax:

```
switch (controlling_expression){
        case case_label:
                statements
        break;
        case case_label:
                statements
        default:
        statements for default

}
```

java 12 allows string statement in switch

RANDOM NUMBER GENERATOR

Class that will generator random number
Random random = new Random() - object of the random class
int number = random.nextInt(10);

0, 1,2….9, it generates 0-9 for 10

STRINGS

String is not primitive data types
we call them string objects
we always use double quotes for strings " "
add two string together with +
there is a difference between "" and " "
"" together called "empty string" there is no character
you can not make String manipulation on chars

Declaring and initializing String

String word = null;
String word = "";

Index : position of a characters

always starts from zero
spaces also characters

String methods

Method - actions done to objects
behaviors actions functions
Strings are objects
methods can be used with objects

some kind of actions that objects uses
important part of methods

- what it takes
- what it returns

public static int addNumber (int number1, int number2)
method name arguments
int - return
addNumber - method
int number1 and int number2 - what it takes

char c = str.charAt(0);

str - object
charAt - method
(0) - what it takes
` 0` is a argument - is a information that method takes

str.charAt() - returns a value

String s = "Java"
String s1 = "is fun"
String s3 =""
String s4 = "Java is fun"

charAt()
takes - index number
return - char
Ex: s.charAt(0) - returns "J"

equals ()
compare string values
takes - string
returns - boolean

Ex: s.equals(s1) - returns "false"
Ex: s.equals("cybertek") - returns "false"

equalsIgnorecase ()
compare string values with ignoring cases
takes - string
returns - boolean

String str = "Java"
str.equalsIgnorecase("java") - true

length()
how long the word is
takes - nothing
returns - int

Ex: s.length() - returns 4 // java
s1.length() - returns 6 // is fun counting character

(s.length() == s1.length()) - comparison returns false

isEmpty()
check the string has character or not is it empty or not?
takes - nothing
returns - boolean

ex. s3.isEmpty() - returns true
s2.isEmpty() - returns false

contains()
checks if the provided string is inside of checked string or not. if they are exactly the same it contains - true

takes - a string
returns - boolean

Ex: s4.contains (s) - returns true because both have "java"
s4.contains("Java") - true

indexOf () opposite of charAt() method
looking for argument string first occurrence position number
-1 not found
takes - a string
returns - int
String str = "Java"
str.indexOf("va") -> 2
str.indexOf("a") -> 1 first occurrence
str.indexOf("vo") -> -1 not found

lastindexOf()
looking for argument string last occurrence position number
-1 not found
takes - a string
returns - int

String str = "Java"
str.lastindexOf("a") -> 3 last occurrence of string

The following methods will need to assigned to a new variable
or reassigned to the existing one
or use the result right away

toLowerCase()
changes the string to lowercase
takes - String
returns - string

String str = "Java"
str = str.toLowercase(); reassigned again you need to store it

toUpperCase()
changes the string to lowercase
takes - string
returns - string

String str = "Java"
str = str.toLowercase(); reassigned again you need to store it
String lower = str.toLowercase(); assigned to new variable

replace (old char, new char)
str.replace ('a','o')

replace (old charsquence, new charsquence)
we have overloaded method for multiple letters
str.replace ("java" , "ruby")

*deleting spaces with replace
str.replace (" " , "");

give back a string with every occurrence of the old character
takes - two chars
returns - string
Ex. s.replace('a','o')
Jovo

substring (start index number) you will think "rest" not "cut"
take- integer
returns - string

String str = "Java"

str.substring (2) - "va"
Overloaded substring
substring (start index number , end index number)
take - two index numbers second number outside of it
returns - string in that range "not include" the last index

String str = "Javaisfun"
str.substring (3, 8) - "aisfu" 8 is not included

trim ()
removes any white spaces before and after the string. Spaces between words does not deleted.
takes - nothing
returns - string
String str = " Java "
str = str.trim(); str will be Java without spaces

startsWith()
characters of sequence to be checked for
true if the checked string begins with the parameter string

takes - string
returns - boolean

String str = "Java";
str.startsWith ("Ja"); -> true but it should start with it

endsWith()
characters of sequence to be checked for
true if the checked string begins with the parameter string

takes - string - checked one
returns - boolean

String str = "Java";
str.endsWith ("va"); -> true but it should ends with it

compareTo ()
lexicographically (starts with numbers then alphabetically
uppercase and lowercases)
compare strings which comes first in lexicographically order
takes - a string
returns - integer number
negative number:- if obj comes first lexi order
positive number+: if argument comes first on lexi order
zero. if they are equal
String word = "Java";
String word2 = "James";
word.compareTo(word2);
(checked one) word2= passed/argument
word = checked one/object

Wrapper Class not in String Class
isUpperCase();
takes char
return boolean

isLowerCase();
takes char
return boolean

Convert from String to int

Integer.valueOf(String str) - returns integer value of string
Double.valueOf(String str) - returns double value of string

This string must only contains numbers, otherwise there will be an error

Convert from Char to String

```
char c='S';
String s=String.valueOf(c);
```

char i = s.charAt(0);
int number = i; i=67

char directly turn in to ASCII number

Escape characters

if you want to put quotes in the string
you use back slash

\". " "
\'
\\. backslash
\n new line. goto beginning of the next line
\r. carriage return. goto the beginning of current line
\t adds whitespace to the beginning of next tab

*how to always get to the last index of a string
String word = "we should take a trip";
int last = word.length()-1;
char lastchar = word.charAt(last)

Chaining methods

You can use methods together
That's should be the same variable to use multiple methods
word.toLowerCase().replace('a','o');

LOOPS

Programming concept which allows you to execute certain set of block of code/statements within the given condition.

syntax

```
for (int i =0 ; i<5 ; i++){
//statement will executed multiple times
}
```

int i = 0 -> initialization: happens only once in the beginning
i<5 -> termination condition: when it should stop running
this part is false the loop will end
i++ -> update : increasing to decreasing a value. You can write any type update like (i += 2)
iteration : one time execution of the code inside the loop

if you want to write the i outside of loop. it is not going to work because it is local variable! you can use i for another loop because outside of the loop.
Each parts of the for loop is optional but as a tester you need to put all of them.

infinite loops:

concept when the loop does not have a stopping point or when the termination condition is never met.

```
for ( ; ; ) { infinite loop
for (int i = 0 ; i>-1 ;) { infinite loop
```

break; this is the keyword used within the loop to stop the loop even-tough termination not come yet.
continue; skip the execution of the rest of the iteration
it goes to the increment and skip the loop iteration
means don't run anything under it come to update.
simply skip that condition.

break: will stop the loop completely
continue: skip the rest of the iteration and continue to run as normal.

Loop with Strings:

How many times should be loop a String?
    > length times
How can we read each character of a string with a loop?
    >charAt(i)

if you want to change char to String.
String s = word.charAt(i)+"" ; with double citation at the end

WHILE LOOP
for loop : we set exact iteration when we use it
while : amount of iteration will be dynamic. it will be mostly dependent from different statements

syntax
while (boolean statement){
//statements
}
we can pass any expression inside of the parenthesis of while as long as returns boolean statement is false.

BREAK AND CONTINUE CAN BE USED IN ALL OF THE LOOPS

DO WHILE

similar to while loop but the do while loop will execute the given statement at least one time then check the boolean condition to evaluate if the loop should continue

syntax

```
do
//statements

}while(boolean);
```

1- execute the statement
2-check the condition
if true - execute again
if false- exit the loop

for: when we need to control the amount of iteration we know how many times it is running
while : when the iteration is dynamic = we don't know how many time will be run, might be infinite time until the condition is true.
do while : similar to while loop but always one execution before the condition is checked

NESTED LOOPS

nested loops are a loop inside of another loop

Syntax:

```
for (int i= 0; i<5; i++){
     for (int j=0; j<5 ; j++){
       //statements
    }
}
```

ARRAYS

Arrays stores collection of elements of **the same data type.**
first type of collection we will use
We use collections to collect same type of data
import.java.util.Arrays is their class
Arrays can store primitive data types and object data types
primitives: int [] array;
object: String [] array;

Arrays are also objects
this means that Arrays should be able to hold other arrays

Ex:
String name1 ="james"
String name2 = "joe"
String name3 ="mary"

datatype [ ] arrayName = new datatype[arraysize];
String [ ] employerNames = new String[10];

- datatypes can be primitive data types or objects
- [ ] : syntax that show this is an array
- variablename: follow the same variables naming rules
- new: keyword to make object
- [size] : an integer that tells how many things the array will store

- Each part of an array is called an element.
- Every element has an index.
- When you make a new String array all elements are null
- When you make a new int array

1.How to store information into an array

Syntax;
arrayname[index] = value;
employerName [10] = "joe"
employerName [10] = name2;

*The value being stored into the array should be same data type of the array being storing into.

2.How to read information into an array

System.**out**.println (employerName [10]);

employerName[10].charAt(0); you can think this array reading is string value you can do anything to it with methods.

nums[3] = nums[2] + nums[1]; // adding values from their positions and putting into num[3] container

3.Printing an array

*System.**out**.println (nums); // if you want to print array name itself you will get [7&89098 kind of position number

*Arrays.toString(nums); // if you want to see array values together use toString () method

how many iteration you will do for array you need to use the arrayName.length

for dynamic storing and reading we are using for loops

length() vs length
length()- used to get length of the string
length- used to get length of the array

Alternative way to create an array while giving the initial information

int [] mums = { 2,3,4,5,6}; - initialization list

ARRAY CLASS
when you are using Array class from java.util package for common array operations

import java.util.Arrays;

- java.lang.Object
  - java.util.Arrays

public class **Arrays**
extends Object

Arrays.sort(arrayName);
it is sorting the elements lexiograhically smallest number to largest numbers
special character - numbers - uppercase - lowercase

Arrays.sort (nums,4,9); partial sort sorting 4 to 8

Arrays.binarySearch(arrayName, element);
allows us to search for an element in an array. but array
HAS TO BE SORTED.

Arrays.copyOfRange(original array , from , to)

Arrays.equals(arr1,arr2);

Possible results:
if the element is found it will return the index of the element

if the elect is not in your array it will return you negative number position if the number would have been if there was in your array (- that position+1)

if array is not sorted the result is not predictable

STRING MANIPULATION METHODS

split()
used on a string and converts it into string array based on array based on the given characters

takes : String
Return : String[]

String str = "Java is programming language"
String [] regular = str.split(" ");
if we split by space
we split the sentence word by word and we stored in the array
takes away the spaces

(*) Other methods that could be helpful

-- Arrays.equals()
-> Will check if two arrays have all the same elements
> Takes: two arrays
> Returns: boolean

-- Arrays.copyOf()

-> Will make a copy of the array you have, and allow you to increase the size of the new array

> Takes: array and int (size of the new array)

> Returns: an array

Collections: One object storing multiple datatypes

Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).
This loop can only used with collections (you can use any type of array string array, integer array or any array)

2D 3D MULTIDIMENSIONAL ARRAYS

Arrays are also objects
This means that Arrays should be able to hold other arrays
Syntax
data type [ ] [ ] varName = new datatype [how many array] [how many element in inner array];

size of inner array is optional takes any size of array in same type even though inner size declared it takes any size of array

after declaration arrays will be "null" because arrays are objects

varName [position number of the array]= {1,2,3,4,4};

writing your array
System.***out***.println(Arrays.*deepToString*(groups));

Syntax for initializing
int [][] groups = {{1,2,3,3}, {5,6,7,7}, {8,8,8,8}};

3 int arrays initialized
every array has 4 element

we will use nested loop for reading writing elements of arrays
you can use nested each loop, too

EACH LOOP
This loop is made to through each element in the collection
This loop READ only, which means only need to go through the elements and check them you can use it

syntax
for (dataTypeOfElements nameOfElements : collection name){

}
nameOfElements -> int num = numbers[i]
When ever you create a CLASS you create an OBJECT!!!

METHODS
What are methods;
Set of java statements in one code block
purpose of a method do some behavior and actions
method have some function

1. Creating methods
when you create a utility classes you are create your own methods.
2. Using methods
when you use any method in your programing.
Arrays.sort(); using someone's method

Ex. String manipulation methods : length() , indexOf()

Method signatures has
access modifier (optional)
return type
method name
parameter / delimiter

Methods do not store data
Actual method does the action but does not keep the data

Why we write methods?
if there is an action that needs to be done many times then the method is created and called anywhere it is needed

Note: to use the method action you have to call the method in the main method

There is two type of method:
void
return

Most important part of methods
(when you look a method always ask yourself)
- what it takes
- what it return
- what kind of return data type

rules for making methods
1. never create a method inside of another code
this mean you can not create a method in main method too
but all methods should be in inside of a class

2. give the methods a meaningful name

follow camel case
never start with uppercase or numbers

3. one class can not have two methods with the same name
4. we create separate java.class classes then we will call from that files

VOID METHODS

method does not return a value. only does an action
you can use "return" keyword for terminate the void method, even though it is void it just

Ex. Arrays.sort() is a void method does not return anything
notice we do not store anything anywhere

```
public static void sayHi (){    //public static then return type:void then name of the method:sayHi

    System.out.println("Hi!");
}
```

implementation: my implementation means how i write the code my way of solving it

data transfer: "method parameters"

passing some information to the methods that can be used in the methods

Syntax:

```
public static void printName(String name){

//statements
```

}
in this case without a string your method is not going to work
multiple data transfer:
accepting multiple data in the parameters

```
public static void goTo (String origin, String newLocation){

//statements
}
```

VARARGS
methodName (1,2,3,4,4,6)
for varargs you can passed numbers in the method body

int ... arr - three dot in the middle you can put

int ... arr is called varargs

when you use varargs you need to put them at the
end of the parameter arguments

findNegative (int num, int ... arrs)

RETURN TYPE METHODS
method that give a single value back to the user. the method gives one information. the value given back should match the datatype that is provided in method declaration
— instead of the void keyword we need to have the "datatype" we expect to method to give

— keyword: return

the return type method needs to have return keyword to give a value back without this return statement there will be a syntax error
error: return statement value needs to match the datatype given as the return type of the method

—as soon as the code comes to return statement exit from method return should be and.
with return statement is the method is end
code after return statement is "unreachable"

— multiple returns: but you can have multiple return statements in the same method block you can control via switch or if statements.

—you can put parameters in the return type methods

this method returns us number 3
— syntax

```
public static int getThree(){
    return 3;

}
```

public static + datatype + methodName()

only writing
getThree(); in the main method you will not see anything on the console or anywhere it not going to effect anything else because you need to catch it.

int a = getThree(); with writing "int a= " basically you catch the return when you invoke your method
the method returns integer goes in integer

OVERLOADING METHODS

Having multiple versions of same method in one class
this allows you to have a method with the SAME NAME, but different implementations

Ex: substring() println()

—Method signature :
—name of the method, numbers of parameters type of parameters
—return type is NOT the part of signature
- you use method signature to change

public static void draw(String color) — this is method signature

*RULES OF OVERLOADING A METHOD
1. give the same name to a method
2. change the number of the parameters
3. or change the data type

public static void draw(String color)

public static void draw(int size) - successfully overloaded method we change the type of the data

public static void draw(String color1,String color1) - we add the new String

public static void draw(String color1, int size) - you can put any type
public static void draw( ) - without parameter this overloaded ,too!

if you successfully overloaded the method then you can change the return type or you can keep the return same.

Does change the return type is overload the method?
NO
return type is not the part of the signature, in this case you also confused the java

Datatype parameter promotion

byte < short < int < long < float < double

Even though data type is not there it is using the biggest closest (it promotes the data type to biggest and closest) data overloaded method.

VARARGS
Using varargs allows you to pass an array as the argument without having to create an array first.
you can put array directly to the parameter part

```
public static void main(String[] args){

printArray(3,2,4,5,5);

 }

 public static void printArray(int ... arr){
```

}

you can not have two varargs as a parameter in the same method it should be the last parameter in the method if you put varargs first java will confuse which is the number which one is array.

OOP (OBJECT ORIENTED PROGRAMMING)

Class vs Object

1. Class: IS A BLUEPRINT - TEMPLATE - STRUCTURE OF AN OBJECT. It is a template that specifies ATTRIBUTES - (instance variables) and BEHAVIORS (methods) OF AN OBJECT. The class carry information about an object will be created by itself.
2. Object: an object that got created by the class

multiple objects can be created by one class.

Mouse m1 = new Mouse();

Mouse - Class Name - data type information about the object
m1 - reference variable
new - keyword that creates the objects
Mouse() - Constructor helps creating object

Java allows us to create objects in computer using classes.
Java puts the objects in a class. Creating class means you are creating a certain type. Class is a blueprint of an object.

you will use classes for creating an object. Only one class you will set the rules in a class then you can create as many as objects you want like blueprint. you can create as many as houses with one blueprint plan.

Datatypes in Java

1. Primitive data type - only carry one value int i=6; Primitive data types are able to store ONLY one piece of information.

primitive
- only one information

2. Object data type - we can store or create information in an object. Object data types are able to store many pieces information. Objects also can store behaviors of objects.

object
- attributes (information)
- behaviors

Creating an object (instantiating an object)

1. We have to create a class and set the rules for it.
2. We always use "new" keyword every time when we create an objects

```
Flag f1 = new Flag();
```

3. Instantiate - create an object
4. Object - instance
5. Create an object from it.
6. reference name (f1) is pointing to object

String is an object.
String name = "james";

string - object type
"james"- value

I can use methods on this String . name.indexOf("m");

Class has 2 things

1. Attributes = Data fields = instance variables = properties

2. Behaviors = Methods
Dog is our OBJECT TYPE. (like String)

```
public class Dog {

String name;
int age;
String color; OBJECT ATTRIBUTES = FIELDS= INSTANCE VARIABLES
double weight;
String breed;

 // you don't put any values to this instance variables. It will be
default same value for any object you created.
String name "max";
All Dog type objects will have "max" as a name. not a good idea!

public void bark()
          {
           System.out.println ("Dog is barking");
           }}
```

```
Flag f1 = new Flag();  // objectType objectName = new constructor();

f1.country = "USA";   //objectName.instanceVariable = value;
f1.size = 5;
f1.material="fabric";
f1.color="red/white";

System.out.println( f1.country );    //USA
System.out.println( f1.material );  //fabric
System.out.println( f1.size );         //5
System.out.println( f1.color );       // red/white
```

we can create as many as flags with Flag class behaviors and attributes
f1 for USA flag
f2 for Turkey flag etc.

**By default when we have local variable inside the method same with instance variable it uses the local variable in the method. Java always prioritize the local variable.
we need to use THIS keyword for telling that we use the instance variable. this.name -> always point to instance variable.

CONSTRUCTOR ()
constructor with no args
constructor with args
constructor overloading
constructor chaining
constructor is not a method acts like it

What is constructor?

```
public Sky(){ // this is the constructor it is actually hidden


}
```

- it is the special method that is used to create an object
- it not a method!
- it MUST have same name with the class
- it does NOT have return type

- every time when we create an object we use "new" key word and constructor. Java allocates-reserve the memory for the objects, java looks for a constructor next to new keyword.
- you don't need to create a constructor, java creates for you by default constructor.
- default constructor is hidden and it does not take any arguments.
- however we can create a constructor that takes some arguments to set initial values for newly created object.
- you can modify your constructor
- you can have multiple constructors in one class. constructor overloading

Sky sky = **new** Sky();

className referenceName = new Constructor();

referenceName is not a actual object
Actual object is [new Sky();]
it points to object

What kind of code is written inside the constructor?

- Most common use of constructors to set initial state of object. if you create-change constructor by yourself NO MORE default constructor from java unless you write the default one otherwise you will lost the default one. you will get initial values as parameters in the constructor you created.
- you need to set the values in your new constructor.

Constructor overloading?

- while you create an object you pass and set the initial values .

```
public Sky(String colorCode, int visibleStars)
Sky sky2 = new Sky("red", 500);
```

Checking objects is null?

if (f2.country == null) checking the objects is null or not

Constructor chaining?

we call constructor in another constructor. it is called constructor chaining.

```
public Item(String name, int price){
        this.name=name;
    this.price=price;

  }

  public Item(String name, int price, int size){   //overloaded constructor

    this(name, price);   // chaining
    this.size=size;
  }
```

this vs. this()?

this this refers to instance variable
this() refers to constructor chaining

Using reference variables and it's methods?

```
FaceBook obj1 = new FaceBook ();
```

FaceBook obj2 = new FaceBook ();

obj1.sendFriendRequest(obj2);

sendFriendRequest works on obj 1 and the method uses the parameter obj2.

in class when you create the method

obj1 is always mention with "this". this.name means obj1 name

if you created an object belongs to the class (with the instance variables)

Author author = new Author();

you can assign it like that

this.author.name = author;

ARRAYLIST

arrays

- represented by []
- multiple value with same data type (int array, String array etc.)
- fixed size, can not add or take out item, can not be changed
- store primitives and objects
- you can not print the array directly without method
- array is ordered and can be accessed by index.
- arrays allows duplicates (you can put 5 in different boxes)

Collection Framework

comes with JDK

built in features to work with data
provide different type of data structures
- List, Map, Queue, Set

ArrayList Class

import java.util.ArrayList;
import java.util.*

- resizable
- only hold OBJECTS - can not store primitives. you need to create an object to put inside in ArrayList

```
Item product = new Item(name, price);

this.items.add(product); //we added to object to the arraylist of items
```

- you need to use Wrapper classes for storing primitives as an objects
- ordered collection
- allow duplicate
- storing any types (one type recommended)
- can be printed directly

Syntax

```
ArrayList <String> names = new ArrayList<> ();
```

ArrayList methods-

there are many methods that are provided by the Arraylist so we can manipulate the data with them

add(Type element) adds end of the list, it adds the element at the end by default. if you give index adds to specified index

get(int index) returns the element in specified index

```
// create an arraylist to store customer objects
ArrayList<Customer> customerList = new ArrayList<Customer>(

customerList.get(index).getAccountOwnername();
```

items.get(i).**name // items is arraylist get(i) index you get name is the name of the object loop through it**

indexOutOfBoundException if you give the index does not exist

remove(int index) removes element from that index , if it does not find it that index does not do anything. When the element is removed index are shifting. we are giving the object index to remove.

contains(Object element) boolean return true-false

indexOf(Object element) gives the first occurrence index of the element -1 if it does not exist

clear() clears the array makes the array[0]

size() gives size of the array

isEmpty() boolean return

set(int index, Object element) puts the value to specified index. when yo set index are shifting.

explain this and this() interview question

how to terminate the method?

with return;
even if it is not void type.

Primitive Wrapper Classes

Wrapper Classes comes with JDK for creating an Object
that needs to represent primitive types.

Each primitive datatype has it's own corresponding Wrapper Classes

Why we use wrapper classes?

Because collections does not take primitives they take only objects.

objects

- objects can be created from classes
- Classes can have field method
- need bigger memory space

primitives

- no object can be created
- there can be no fields and methods
- light weight requires less space

boolean	Boolean
char	Character
int	Integer
byte	Byte
short	Short
long	Long

float Float
double Double

They have two constructor one taking String the other one takes Boolean

Integer i1 = new Integer (1202);
Integer i1 = new Integer ("1202");

primitive -> object BOXING without new keyword autoboxing

object -> primitive UNBOXING

*java programmer does not have to explicitly cast it, it happens automatically.

(*) Custom Class Members

1) Instance variables
- Information that belong to each Object which is created from the specific class
- In order use instance variables you MUST have an object

2) Methods
- Behaviors that belong to each Object

3) Constructors
- Special kind of method that creates an object

- STATIC MEMBER
4) Static members
Variables and methods can be static. They can be called by class name.
Static members are belongs to a class doesn't belongs to an object.

This means you don't need to create an object to use static members.

You can access the **static** member without first creating a class instance. Objects are sharing the static variable value.

main store
store1
store2
in any store static variable numberOfComputers changes the other ones are changing, too.

```
class Walmart{
      String address;      -instance variable BELONG TO AN OBJECT!
      double size;         -instance variable
      int numOfEmployees;  -instance variable
      String type;         -instance variable
      static String CEO;   -static MEMBER - only one CEO for
                            Walmart should be shared with the locations
                            BELONG TO A CLASS! static member is not an
                            instance variable!

      }

Walmart w1 = new Walmart();   Store1
Walmart w2 = new Walmart();   Store2
Walmart w3 = new Walmart();   Store3
```

Access to static members:

You don't need an object to get an access.
Both object and class itself have an access to static members.

w1.ceo = "Mr.doug";

sout(w3.ceo); // prints Mr. doug. object accessing to static member

How to access using ClassName?
Walmart.ceo = "Jason"; class accessing the static member

Ex. of static method
YOU CAN WRITE CLASS NAME TO CALL THE STATIC METHOD
IF YOU IMPORT CLASS NAME WITH WILD CARD YOU DON'T NEED TO WRITE THE CLASS NAME, TOO!

ClassName.followedByStaticMethod();
Arrays = class name sort() method

Arrays.sort();
Array.toString();

Ex. of non static method
obj1. followedByNonStaticMethods();

nextLine(); from Scanner we always create an Scanner object
trim(); from String class

Utility class methods are all static. you can call it whenever you need it. Anytime you create a utility class for yourself you need to use static variables.

public String toString() { NO STATIC HERE
// we don't have "static" here so it does not belong to a class. you need to have a class object for calling toString() method
Book book1 = new Book ();
book1.toString(); // we need to have an object book1

Static block (static initializers)

we will use it to initialize our static variables.

Syntax:

```
static {
//statements. anything is in should be static variable

}
```

Rules:

- every time when you run your code static block will be the FIRST running block without calling or anything.
- Static block can only have static members
- you can have as many as static block you want. They will run top the bottom.
- you don't want to use static members on constructor or constructor overloading. constructor is belongs to objects, static members belongs to a class. we can not have static constructor
-

IQ: Do you have to assign static members in the block?
No, you might want to use their default values.

IMPORT STATIC

if you want to import static members (VARIABLES AND METHODS) to your class you can import the member instead of the whole class

import static java.util.Arrays.*; it is just importing all static fields in Arrays class BE CAREFUL WITH THE "static" word
BE CAREFUL you need to import static members directly or you need to write wild card after CLASS name

```
import static mypackage.ClassName.sellCar;
import static mypackage.ClassName.*;
```

without static import regular import goes

```
import static mypackage.ClassName;
```

import static utilities.MyCollection.*; we have another class in different package, if you want to call static members you will import like that the YOU CAN USE THEM WITHOUT CLASS NAME.

keyword

you can write sort();

you don't have to write Class Name
Arrays.sort();

for testing

- utilities package: tools for your project. package that will have classes that are helpful for the project
- most methods behave like a tool
- the methods in utility class are static

FINAL

Final keyword is make the variable value final - you can not change it. Final keyword can be applied to variables, methods and classes.

-> final variable:

when "final" keyword is applied to a variable, that variable value can't be reassigned, changed anymore.

Ex:
int i = 5;
int =12;

Ex:
final int i = 5;
int =12; // compiling error

Final variables also known as CONSTANT VARIABLE. used for things that sound't be changed. We usually name them with UPPERCASE letters.

Ex: final double pi = 3.14;
final int weekLOfTheDays = 7;

Local variables vs Instance Variables

Local variables: Declared within the scope of methods or any curly braces. They can be used only within the scope.

Local variables don't have default values. Programmer should initialized them in the scope. we can declare and assign later for local variables

but you need to initialize right away for the instance variable.

Why we can not use instance variables in main method OR static method?
Non - static variables(instance variables)

You can not use instance variables in main method without creating an object or you need to have non static method to use non static instance variable. Instance variables belongs to an object and they are non-static. Static variables belongs to a class.

STATIC VARIABLES BELONGS TO CLASS -> works on static to non- static methods

INSTANCE VARIABLES (NON STATIC) BELONGS TO OBJECT -> instance variables don't work on static methods. You need to create an object to use it.

STATIC METHODS -> It can not be overridden.

Static Methods can access **static** variables without any **objects**

Static methods can be accessed directly in **static** and non-**static methods**.

You can not make the class static

FINAL CLASS - You can not inherit final class. That class can not be the parent of another class. It cannot be inherited.

FINAL METHOD - It can not be overridden but you can overload.

Where to initialize final instance variables?

- option 1: initialize final instance variables right away in the class. Give the value to instance variable when you declare it
- option 2 : initialize the final variables in the constructor. Because instance variables can not be used without creating an object
- option 3: we can initialize final variables in the instance initializers block. you can write any code in initializer block but you can not use code in instance variable level.

MEMORY MANAGEMENT IN JAVA
STACK AND HEAP MEMORY

When we run the program it is happening in RAM(Random Access Memory)

There are two places that Java uses from RAM memory.

1. Stack memory

- primitive data type information

```
int i=5;
boolean check= true;
```

- references

```
Bag b1 = new Bag();
```

b1 is a reference name stored in stack memory. b1 contains only heap address of new Bag(); pointing this Bag object in the heap

- method callings

```
MyCollection.getStringList();
```

When ever your computer stack memory is full gives you "Stack overflow error"

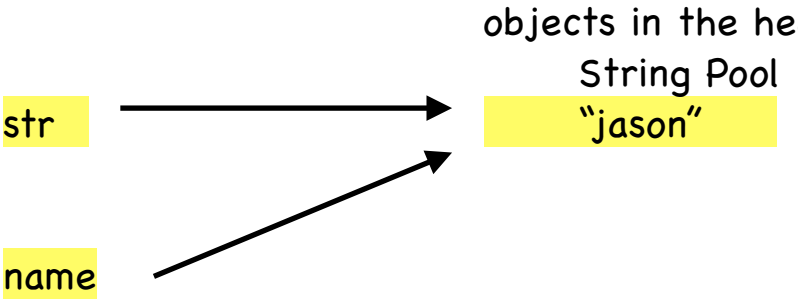
2. Heap memory

All objects are stored in the heap. Strings are objects and combination of characters.

char size = 2 byte = 16 bit.

String name ="jason"; = 80 bits.

String str =" jason"



if you create same object as a string like "jason"
all of the references has the same name pointing to same string object in the heap. Java puts all the unique objects in the String pool. String pool it is a corner in the heap where unique string values are stored.

== vs. .equals()

sout(name == str); comparing the addresses
sout(name.hashCode()); @775784
sout(str.hashCode()); @775784

hashCode is not the memory address! hashCode between address and the reference
hashcode is not the actual memory address but its a link for JVM to fetch the object from a specified location with a complexity of O(1). A hashcode is an integer value that represents the state of the object upon which it was called.

String str2 = new String("jason");
you can have separate jason this way. now the jason is not in the string pool in the heap memory.

== checks if both objects point to the same memory location whereas .**equals()** evaluates to the **comparison of** values **in the** objects.

What is the difference between String object and String Literal?

GARBAGE COLLECTOR
Garbage collector cleans up the memory implicitly.
You can call it by System.gc();

it executes finalize() method for objects with no pointers.
programmers use finalize() method for proper clean up.
Mouse m1 = new Mouse();
Mouse m2 = new Mouse();

m1.color= "White";
sout(m2.color); - will be null because we did not assign any value

Mouse m3 = m2;
if we use this line we can keep the m2 object in m3 reference variable.

m2 = m1;

Both m1 and m2 is pointing to m1 object that's why we lost the m2 object. Garbage collector will clean it up. from heap. m1 gave its reference to m2. m1 and m2 hold the same reference address.

Order of execution?

- First execute static block only once
- Second execute non static block(without creating an object this block will not run, even tough you don't do anything with the object. Every time you create an object it will be executed)
- Third execute constructor every time you create an object

MUTABLE VS IMMUTABLE
We can talk about mutability for only objects not the primitives.
We talk about java classes objects. Usually mutability is not used with custom classes.

Item item = new Item ("pen",3);

Item = object data type

item1 = reference name
new Item("pen", 3) = Object is created, by invoking the constructor

mutable - original object able to be changed. Original object can be changed. Data about this pen can be changed. item is mutable
item1.price = 25; you can change it. you can reassign it.
StringBuilder is mutable

immutable- Original object can not be changed
String are immutable
Date object is immutable
wrapper classes are immutable

when you change reassign your string, it is not the original object anymore.
The original string is losing its reference.

STRINGBUILDER CLASS
mutable version of String objects

syntax :
StringBuilder sb = new StringBuilder("James Bond");

methods :
append("James"); concatination
Takes string or number
append(5);

sb.append (" Bond"); we don't need to assign it the original object changed to "James bond"

deleteCharAt(index);

delete(begin index, end index); same like substring does not take the last one.
reverse();
insert(index, String); it will add the given string into a specified index

1. convert String to StringBuilder;

you can pass the string to a StringBuilder object
StringBuilder sb = new StringBuilder("James Bond");

2. convert StringBuilder object to String
you will use toString() method

String name = "james"; 1

StringBuilder sb = new StringBuilder(name);
sb.reverse();
name= sb.toString(); 2

you can not compare word . equals sb.reverse();
you can not compare StringBuilder object with String that's why you need to use toString()

ACCESS MODIFIERS

Java uses access modifiers concept to give proper access to determine the scope of variables, methods and classes.

Project hierarchy
-> Project (main project)
    -> src (folder that stores codes)
        -> packages (src can have multiple packages)
            -> Book.java (java class file)

Can we use codes of project from another project.
No, we can not use it straight forward but we can package our program as a ".jar" file format and add it to a different project then we will be able to use it.

1. public -> CAN BE ACCESS ANYWHERE WITHIN THE PROJECT
2. private -> CAN BE ACCESS ONLY WITHIN THE CLASS
3. protected-> CAN BE ACCESSED
4. default -> THERE IS NO ACCESS MODIFIER ACCESSIBLE IN THE PACKAGE. SCOPE ONLY WITHIN THE PACKAGE.

default = package protected

* every single variable, method, class or constructor MUST have access modifiers except local variables.
*all 4 access modifiers can be used for variables and methods. Java class can only use public or default.

public Class - if you write only class. the class will not be available in different package. in default java gives package accessibility

Can you make private class?
!!!private Class - NO!!! it is conflicting with the concept

***NO CONNECTION BETWEEN ACCESS MODIFIERS
AND FINAL - STATIC OR IMPORT
importing about using the class
access modifier about scope for member of the class and class itself

***Any class in different package you MUST IMPORT IT

Can a constructor be a private?

Yes. that private specific constructor you van use it in that class.
if you don't want anyone to create an object out side of class from (private) constructor.

What happens if constructor is default?
it is in the reach of package. you can create that object in the package.

***class, variable, method or constructor can be DEFAULT

ENUMS - ENUMERATIONS

we can say it is a datatype will allow us to give all values that a variable could have
you can create your own enum with specific type of data Ex. school grade should have A B C D E F

enums act like a class you can make object of enums
Syntax:
enum nameOfEnum {fields}
fields = all the possible values you can have

enum grades {A,B,C,D,F}

rules:
1. enums CAN NOT BE LOCAL
2. no need semi color
3. naming convention is all uppercase
4. fields are usually name uppercase
5. if don't add access modifier the default one will be private
6. you can take enums as a parameter in your method
7. for example if you have database enums you can connect your data base with enums

OOP IN JAVA
OBJECT ORIENTED PROGRAMMING
treating everything as an object in programming

1. Encapsulation
2. Inheritance
3. Abstraction
4. Polymorphism

ENCAPSULATION
data hiding

- technique of making the variables private in the class
- private variables can not be accessed DIRECTLY outside of the class
- providing access INDIRECTLY to private fields via public methods

achieving encapsulation

1. make your variable private
2. provide public methods to set and to get the values.

SETTERS
setting the values in public method

we have a chance to put a code in setter or getter method

GETTERS
getting the values in public method

****nullPointerException
(CREATE NEW ARRAYLIST IN CONSTRUCTOR)

Array list need an instantiation otherwise you will have nullPointerException
because the array list takes objects you need to assign it through constructor otherwise your program will terminate
if you have overloaded constructor you need to create Arraylist object there, too

INHERITANCE

Inheritance is process when once class acquires the properties of another class
when you want to use the other properties inherits

- methods
- variables

why inheritance?
- reusability
- less redundancy
- better structure
- less code and organized code

!!!No Constructors are inherited - Constructors are belong to a class

How we can inherit?

"extends" keyword

public class CHILD extends PARENT{

public class SUBCLASS extends SUPERCLASS{

public class Dog extends PetSmart{

Rules of inheritance

1. Inheriting class is called CHILD CLASS - SUB CLASS
2. The class passing the inheritance called PARENT CLASS - SUPER CLASS - BASE CLASS
3. Every single class in java is inheriting HAS A PARENT
4. Every class has a parent (object class) unless you specify it with extend. OBJECT CLASS is very first and top of the hierarchy . methods inherited from object like equals(), toString(), hashCode(), clone(). This class does not have a parent. By default every class you created is a child of OBJECT CLASS parents classes are inherited methods and variables to its child class unless we specify the parent. Even tough you extend your parent, object class still your parent of parent ,then your class will inherit from Object class.
5. One java class can only have ONE PARENT
   JAVA DOES NOT SUPPORT MULTIPLE INHERITANCE.
6. ONE PARENT can have multiple of CHILD CLASS
7. Inheritance only happens PARENT TO CHILD. NOT OTHER WAY AROUND.
8. We need to extend the closest class to not miss any variables and methods BETWEEN that hierarchy.

Animal -> Mammal -> Dog -> Husky -> Labrador Husky

IS-A RELATIONSHIP = inheritance "A is a B type of thing"
(is a)way of saying = This object is a type of that object?
IS - A subclass IS-A super class

Dog (is-a) Animal correct
Dog (is-a) App not correct is the inheritance make sense or not.

Animal -> Mammal -> Dog
Dog is a Mammal
Dog is an Animal , too
inherits grandparents methods and variables too.
Logically, for proper Inheritance we should extend the super class that has an (is-a) relationship with a parent class.

CanonCamera extends Camera {
CanonCamera is a Camera. CanonCamera is type of Camera

inheritance vs. importing class

import: (ABOUT PACKAGE) Importing is using classes in different package in your class. If we use class from same package we don't need to import. if the class in the different package you get its methods. Like you use Scanner methods on the Scanner objects YOU NEED A SCANNER OBJECT to use its method. "input" is a **Scanner Object.**
input.nextLine(); this is not an inheritance. with inheritance the child object can use Parents methods and fields.
ArrayList<String> names; //we use Arraylist class
we need to import like that - import java.util.ArrayList;

inheritance :(ABOUT PROPERTIES) If you inherit you will use all the methods and fields of parent class on your object. The **Child Object** can use Parents methods and fields. You can use your parents methods on your object.
Even tough you inherit a class from different package you need to import them.

Object Class

Parent of every class. It does not have any parent. Invisible.

All the methods including arrays are coming from object class.

java.lang.Object

What methods and variables are inherited?

Child class inherits accessible properties ONLY from Parent class public, default and protected available in the subclasses

protected

Protected variables and methods reachable from different package and different package as well ONLY from subclasses.

Ex.

```
protected String color;
// This variable available in the subclasses of Laptop  which is Dell MacBook Lenovo
```

private

```
private String color;
// This variable IS NOT available from different package or not in the subclasses of Laptop  which is Dell MacBook Lenovo
```

this(ABOUT CURRENT CLASS)

this -> reference to a current object, also for name crushing local and instance to show you are talking about instance variable

this() -> referencing to the current class constructor.

this() MUST BE DECLARED AS A FIRST LINE OF STATEMENTS.

super(ABOUT SUPER CLASS)

super -> reference to super class object.

```
super.memorySize = 45; // we access to parent class with super keyword variable and assign it to 45
```

super() -> calling super class constructor

super() MUST BE DECLARED AS A FIRST LINE OF STATEMENTS.
instead of super(), this() can be used. How super class object gets created then?

- this() will use constructor chaining.
- you can not use super() and this() together conflict happens!!!

be careful when you overload the constructor in the parent. Child classes will complain!!

SUBCLASSES NEED A DEFAULT PARENT CONSTRUCTOR. there will be conflict because child objects calls the parent default constructor every time object created.
Two solutions

- keep default parent constructor
- make it in the child class write super(give the parameter the parent class wants) under child class constructor

YOU CAN NOT USE NON-STATIC VARIABLES AND METHODS WITHOUT USING AN OBJECT.
ANY CLASS IN INHERITANCE PROCESS CREATE AN OBJECT FROM CHILD CLASS, SAME TIME PARENT CLASS OBJECT WILL BE CREATED BY JAVA.
when you create a Dell (subclass) object behind the seen you create Laptop (superclass) object, too.

```
public MacBook(){.  // default constructor is like that with super calling parent class
   super();
}
```

WHEN YOU EXTEND JAVA HAVING DEFAULT CONSTRUCTOR LIKE THAT IT IS INVISIBLE.when macBook object created super class

constructor runs, too. first it goes to laptop then MacBook constructor.

MacBook macbook = new MacBook();
3 object created
1- MacBook class object
2- Laptop class object
3- Object class object

INHERITANCE STATIC MEMBERS

They also inherited as well if they are not private. we can use parents static members in child classes. Static is shared with all subclasses. Both child and parent point the same static member. Static blocks are not inherited.?

DATA HIDING

When we have our own static variable in the child class, parent class will be hidden (data hiding)
We can have same name variable of parent class, in the child class, when you assign them in the child class java prefer the child class variable to assign. the parents data will not change. even tough it is STATIC variable(static variables shared but not in inheritance). The class hides the data from parent class data, means does not affect the parent's data. The parent and child have their own variables in this way.

METHOD HIDING

if we want to change our non static methods is overriding. We are changing the implementation of it. when you want to do same thing for static methods this is called METHOD HIDING. Your parent static method will be hidden. Same overriding rules applies for the static methods hiding. We don't use annotations for that.

What is inheritance?

using classes properties

when we extend the class we are getting all the properties of parent class unless they are private

Why inheritance?

less redundancy usability

how inheritance work?

extends keyword

how classes structured in inheritance process?

all classes has a parent top class is object class

what is super() keyword?

class constructor has super() keyword java creates parent class object as a first object . that s why super should be first line

super vs super()?

super keyword refers to the object of super class

this this() vs super super()?

extends keyword

Data hiding

when we have our own static variable in the child class, parent class will be hidden.

since static variable belong to a class it doesn't matter you can access it from everywhere when you have same name variable java doesn't allow duplicates the variable will be the child's own variable

Inheritance accessibility

is - a relationship

child should be a type of parent class helps us do a proper inheritance

object class

inheritance vs import

import about package, inheriting about properties only way to read data from i need to use scanner (i just want to use i don't need its properties) they are different subjects they are not comparable

data hiding

method hiding
method overriding

METHOD OVERRIDING

We don't have to use all inherited properties.
Some properties is useful some of them not.
Sometimes inherited method might not doing the action what we want. We can override the method.

Method overriding: Providing with new implementation for inherited method in the child class.
@Override —> called "annotation"
Annotations can be used for classes, variables and methods

They are used to give a certain meaning to classes, methods and variables.
we use @Override annotations to override a parent method in the child class
@Override annotation is optional but highly recommended. It will helps us to show-do proper overriding. It will complain when you are not proper overriding.

Overriding rules

1. Method name MUST be same
2. Parameters MUST be same
3. Access modifier MUST be same or more accessible. Changing it more restricted is not allowed.
4. Return type MUST be same or covariant. Co-variant -> child of a class. Covariant means same type of object. Huskey is a covariant of Dog. not vice versa. it should be same level or lower. Primitive data types don't have covariants. we use it for object. If it is VOID it must be VOID. there is no covariant of it. if the parent class method is PRIMITIVE return the child class override methods should be same type PRIMITIVE return.

5. When you override VOID return method you must have a VOID return in the overriding.

Method overloading and Method overriding

1. Method overloading happens within the same class, same name with different parameters. They are sharing same name but technically they are different than each other.
2. Method overriding happens in Inheritance, same name, same parameters. Overridden are connected to each other and both are the same methods but the implementation is different.

Can we override static methods?

No. It can not be overridden. However we can change the implementation without using @Override annotation. this process called method hiding.

ABSTRACTION

Abstract class is like a blueprint all child classes have to have that plan from the abstract class. You can have abstraction with inheritance. Abstract classes meant to be extended.

Abstract methods does not have their implementation (codes). We override to parent class methods with OWN implementation, we don't need to see parents methods abstraction solves that problem.

Abstraction:

- Focusing on the "what it does" rather than focusing on "How it does".
- Focusing on the high level methods, NOT focusing on the implementation of it

Example:

Browser: We know browsers MUST have launch, browse, addTab, switchTab, closeSpecificTab, closeBrowser etc. methods. These methods are the methods makes the browser as browser. In abstraction we ONLY declare the methods, but we don't write any

code for it. Because it is abstract. The child class override the abstract parent methods anyway.
Chrome, Firefox, Safari browsers have their own style of coding for launch, browse, addTab, switchTab, closeSpecificTab, closeBrowser methods and they are override on the parent methods.

1. Abstract class:

- We add "abstract" keyword to class declaration.
- You cannot instantiate an object from abstract class. Because it might have abstract methods.
- The abstract class may or may not have abstract method
- The abstract class can have regular methods.
- The abstract class can have public/static/final/ instance variables.
- Abstraction doesn't have anything to do with variables.
- Static methods can not be abstract. Because you can not override them.
- Abstract classes MEANT TO BE EXTENDED by other classes.
- The abstract class can extend any other class. It already extends Object class.
- The abstract class can only have non-abstract methods but it does not make sense. if you make on a class as an abstract you should include at least one abstract method in it. Even tough the abstract class don't have an abstract method you can not create an object from abstract class.
- The child class that is extending to the abstract class is called Concrete Class. Ex: Chrome and Safari is concrete classes.
- The abstract class can override non abstract methods.Because abstract classes can have a method with body/implementation as well.
- Abstract classes CAN NOT BE FINAL. Abstract class meant to be extended.

- Abstract class have a constructor. Because of the chaining of object class, it has a constructor. It needs to pass the super(); reference in its constructor to extending to the parent.
- Abstract class can extend abstract class. Even tough you don't override the abstract methods of parents it will compile.
- Abstract class only extend ONE abstract class.

```
public abstract class Vehicle(){
}
```

Abstract class Browser now CAN have abstract methods.

- Vehicle vehicle = new Vehicle(); IS NOT ALLOWED.

Abstract method: a method that doesn't have an implementation in the abstract class.

- We add abstract keyword for the abstract method.
- We don't use curly braces we finish method statement with ; semicolon.
- Abstract class can have return like int, String..
- You can not instantiate an object from abstract class. It does not make sense to create an object from abstract class.
- You MUST implement(override) ALL OF THE ABSTRACT METHODS OF THE PARENTS in the child of abstract class.
- You CAN NOT have abstract method in non abstract class. because you can not create an object from abstract class.
- Static methods can not be abstract. Because you can not override them.
- Abstract methods CAN NOT BE FINAL.

2. Interface:

Additional abstract methods that don't belong to all classes. You can choose your classes as an interface. Ex: Browser abstract classes you have to have launch() method. but for interface you might not have the method (it is optional-extra) being a browser. so that we create an interface class and the public abstract method in it. It is used to provide total abstraction. A class that implement interface must implement all the methods declared in the interface.

To implement interface use **implements** keyword.

Rules of Interface

- All of the methods in the interface are by default "public abstract", you don't have to write.
- 3 types of methods in interface
- public abstract method (by default) Ex:void turnOn(); - no body
- regular static methods - have a body
- default methods - different default for interface . Regular method in the interface with implementation. this "default"it is not an access modifier! it is a keyword in interface
- you can have variables in the interface if you need to but they need to be final setting. by default all variables "public static final" they need to always be initialized. They need to be capitalized. int AGE=0;
- we can have MULTIPLE INTERFACES AT ONCE
- When you implement in your parent class your child automatically implements the interfaces that the parent class implements. Better practice child class gets the additional implementations from interfaces.
- Ex: Car class has accelerate(), stop() methods
- Lexus (child of a Car class) implements interfaces camera() and blueTooth() as an additional methods. You need to write your own implementations in the Lexus class.

- if you have static method in interface, you can call them with its class name in the child class. STATIC METHODS CAN NOT BE OVERRIDDEN.
- You have to override all methods in the interface to implement
- Interface can EXTEND only to other interfaces not the classes.
- One interface can EXTEND to many interfaces

*List is a interface extends Collections in Java Library

```
List<String> list = new LinkedList<String>();
```

Naming conventions for interfaces

- ends with - able
- can be an action name
- is a relationship
- Ex: Huskey is a Dog Huskey is a Trainable
- Trainable is interface additional feature

```
public class Huskey extends Dog implements Trainable{
```

Huskey has everything from Dog and everything from Trainable

Trainable is interface additional feature

IQ: Can interface be a parent?

Interface can extend only to other interfaces not the classes.
One interface can extend to many interfaces but we don't have parent child relations in the interfaces. When you implement you get the additional methods.

```
public interface GoodFood extends Food, Eatable{
```

we can have multiple extends different than regular extends in inheritance

GoodFood inheriting all of the properties from Food and Eatable