OOP - Java

apsara

Date: ___/___/___

Module - 4

Core Java - Prachi Godbole

Start - date - 20th April 2022

Keywords :

public, static, null, new, if-else, for, while, boolean, import, break, this, true, void, false, continue, private, case, do, return, switch, final, default.... until now we saw.

Package :

Package is a ~~keyword~~ folder name/ directory name. Whoever write a class in java program (programmer/ developer) ORGANIZE the class files using packages.

Packages are needed to ORGANIZE class files.

There are two types of java programmer

1. Class writer (Who is creating API)
2. Class user (Who is using API)

API : Application program interface ⇒ Library

These are nothing but ready made methods (fn) and classes.

e.g. Class Book : Properties : id, BookName
2 constructors
Getters & setters

Class UserOfBook : Main
Create book using default constructor
set values using setters
Print values using getters

Create a book using parameterised constructor (no need of setter) print values using getters.

Date:

eg. If we have two source Files one.java, planet.java
one.java → 3 classes - One, two, three
planet.java → 2 classes - Earth, Mars
If we compile these two source Files, we will get 5.class
Files → one.class, two.class, three.class
Earth.class & Mars.class.

Interview POV.

X.java (source file)
have classes A, B, C ⟹ A.class, B.class, C.class ⟹ No X.class!!!

Therefore, source file has a life upto compile time.
Source file is lost after javac & then JVM works on .class Files, it is not aware of source Files.

Package keyword is written outside the class. It gives information about where to keep the .class Files of the classes
When compiler reads the package keyword, compiler will create the Folder and copy .class in that Folder
Adding Book class in package : study.basics
⟹ study Folder ⟹ basics Folder → has .class
Adding UserOfMain class in package : study.users
⟹ study Folder ⟹ users folder → has .class
Work space
Src : Source Files → .java files
bin : binary Files → .class Files.

Book is Package class now, it must be accessed using
Package qualified class Name
study.basics.test.Book
Package class name

Packages in notepad:
2 Files in dire → One.java & planet.java
Add package Pack1.pack2.pack3
compile this using javac -d.one.java
This will create pack1 folder in dir
compile this using -d ./bin One.java
This will create pack1 Folder in bin Folder in dir

-d ⇒ option of javac command, it tells where to create the package.

import study.basics.test.Book;
import is a keyword
import is written outside the class
import is followed by package qualified class name.

When compiler reads import keyword, it checks entire File for occurances of Book word & wherever if it Finds Book, it replaces it with package qualified class name [here it replaces Book with study.basics.test.Book] Similar to find & replace.

e.g. When we have to use scanner class in our code, we used import java.util.Scanner;

HW : Calculator Assignment

apsara

Date: ___/___/___

Access Specifiers / scopes in java

Java has 4 scopes & 3 access specifiers

Scope	Access specifier keyword	Visibility
Private	private	private properties/ attributes or private methods or private constructors can be accessed only within class where they are defined.
Package/ default	NA	Package scope class / properties/ method or constructors can be accessed within any class of same package
protected	protected	protected properties can be accessed a) within the class where they are defined b) within the same package c) within any sub class of package.
public	public	Properties can be accessed within a) class where defined b) any class of same package c) sub class in any package d) within any class in any package.

study.scopes/s - Alpha, Beta, Gamma
study.conscope - Delta, theta, Userscope

data types in Java - 8 primitive data type

byte	1 byte	char	2 byte single char value - unico
short	integer 2 byte	boolean	True/ False differs from JVM to JVM
int	4 byte		
long	8 byte		
~~sh~~ Float	Floating point 4 bytes		
double	8 bytes		

There are 8 primitive data types in Java

byte	Integer value	1 byte	byte = 8 bits ∴ 2^8 256/2 → -128 to 127
short	Integer value	2 bytes	16 bits ⇒ 2^{16} ⇒ 65,536/2 -32,768 to 32,767
int	Integer value	4 bytes	32 bits ⇒ 2^{32} ⇒ 4294967296 -2147483648 to -2147483647
long	Integer value	8 bytes	
float	Floating point value	4 bytes	
double	Floating point value	8 bytes	
char	single character value	Unicode character 2 bytes	
boolean	true/False	differs from jvm to jvm	

study.datatypes.Example1

Day - 2
21st April

1. One source file can have only one public class and the name of the file should be same as name of the public class
2. If we want multiple classes in one source file we can have
   a) 0 public class + n non public classes
   b) 1 public class + n non public classes.

User of module. ← communicate — Public class

communicate

non public class (helper)

user will not be able to communicate directly with non public classes

Wrapper classes

Package of System (SOP)
String (main args) is java.lang

~~Ja~~ java.lang package is checked by default, so import is not needed.

Wrapper is a object that holds a primitive type.
In Java.lang, for every primitive type, we have a corresponding wrapper class :

Primitive data type	wrapper class
byte	Byte
short	Short
int	Integer
Float	Float

Meaning of wrapper?

Wrapper is just a class which wraps primitive data type to ~~no~~ be available to made an object

User of module

int x = 20;

making obj of Integer class

Integer i = new Integer(20)

obj

wrapper class

primitive

obj = A1	objects.
obj2 = A2	
chobj = A3	
chobj2 = A4	
Flag = A5	
Flag2 = true (primitive)	

stack

heap

A1, A2, A3, A4, A5 ← wrapped objects.

Need of wrapper?

1. Many library classes don't work with primitives, so we can pass wrappers over there
2. Wrapper classes have many utility/useful methods for that data type

Utility methods: Byte.MIN_VALUE → gives min. value for the wrapper class primitive.

String s = "34.56";
double sum = s + 3.2

But this is not acceptable as s is string so we need to parse it in double

Double.parseDouble(s) → s - string will be converted into a double value.

Character wrapper class

In Character Wrapper class, utility methods isUpperCase (char ch), isLowerCase (char ch), IsDigit etc. are used to check for out input string study.wrapper. Example2.
These utility methods are having return type boolean and give result in terms of

H.W. :- Password Assignment.

String, StringBuffer, StringBuilder

How do we create object of string class?

String s = new String ("Helbw");
// creating string object using new not added in const
String s1 = "Good day";
// creating object using literals Added in Constant Pool

Constant Pool : Collection of strings using literals is maintained one per JVM

Whenever we create object using literal, that string / integer gets added in a table (constant pool) after that, if we create new object of same string, no new object ~~of~~ on heap is created, its reff will point to the earlier same string allready there in Constant pool.

pointing to same object in const pool.

String S_1 = "Hello"; // Added in Const Poo
diff
String S_2 = new String "Hello" // not added
String S_3 = "Hello"; // check in const pool, if there available it will point to it.

study.strings.Example1.

$S_1 == S_2$ will compare the addresses of S_1 & S_2

Advantage : The space used is saved as both the obj similar strings share the same object.

If string is created using literals, then many objects with share it so there is a chance of race condition e.g. may objects can try to change it simultanesly
Therefore, to overcome this problem, java made all strings are Immutable : Once created will not be able to change.

Now, we should think that toUpperCase(), toLowerCase(), Concat() are changing our string but these methods create a copy of it & perform these operations, original created string remain unchanged.
study.strings.Example2

Strings are immutable:

Advantages : i) Space is saved due to constant pool sharing
ii) Due to immutability, shared strings are protected from race condition

Disadvantages : i) Every time a string change operation is done a new is created.!!!
copy of string is created so space complexity increases

```
String s9 = "Numbers are ";
For (int i=0; i<100; i++)
{
    s9 = s9.Concat(" "+i);
}
SOP (s9)
```

here every time a copy of string is created to change in it, so disadvantage.

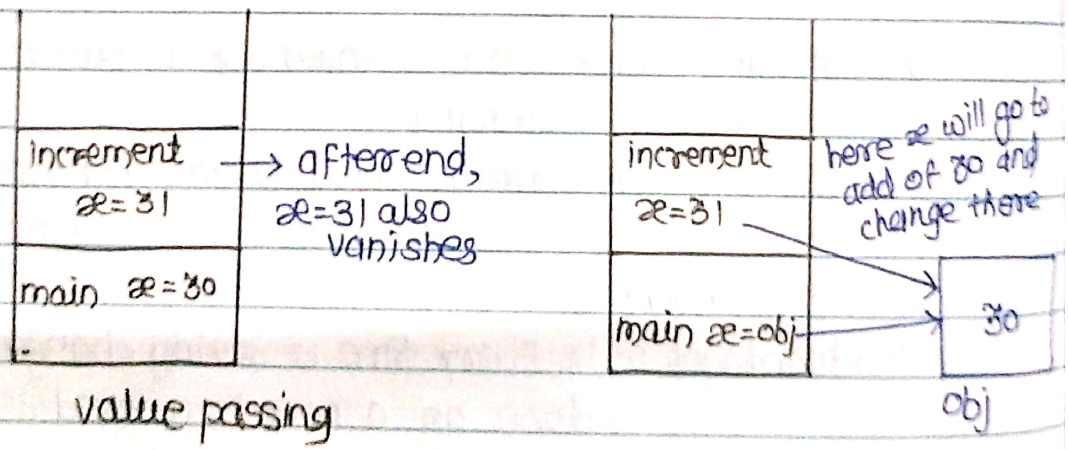
Sometimes, we may need to change a string many times. That time String class is not a good option as too many objects will be created
That time we need Mutable strings so we use two classes from java.lang StringBuffer, StringBuilder
~~study.java~~ study.strings.Example3.

Constant pools are not for strings only it is for wrapper classes also.
study.strings.Example4

Parameter passing in java

If object is passed to the method, the reff (add) is passed and change is get reflect in calling objects parameter

But if variable/value to a method, method can change its value locally and after ~~it~~ method end, it will get removed from stack and changes on variables also get vanisher.



value passing

In Java, parameter are passed by value
For refferences : Addresses are copied between caller & callee
for primitives : values are copied between called and callee.

day - 3
23rd April

Character Encoding
1. ASCII : 1 byte
2. Unicode : 2 bytes

- ASCII encoding uses 1 byte for each character
  1 byte ⟹ 8 bits ⟹ 2^8 → 256 combinations of 0 & 1.

i.e. we can represent 256 characters using ASCII

In unicode encode, 2 bytes for each character
2 byte ⟹ 16 bits ⟹ 2^{16} → 65536 combinations of 0 & 1.

i.e. we can represent 65 K characters using unicode so It has wide range of including characters from any language in the world.

OOps Concepts in java :

What is a class?
class is a design of the object. It includes properties / data members / attributes / fields and behaviours / methods

What is object?
Object is an instance of the class. It is a real time entity based on the design provided by the class

Architect plan → Line plan → class
Building → object

apsara

Date: __/__/__

There are four major pillars of OOP

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

3 minor pillars of OOP
1. Strong typing
2. Persistance
3. Concurrency

1. Abstraction
- Getting only essential things and hiding unnecessary details is called as abstraction
- Abstraction always describes outer behaviour of object
- In console applications when we give call to a function in to the main function, it represents abstraction.

Abstraction is also selection of relevant properties and behaviour in a class.

e.g. While writing a class for patient, we only write properties that are related to medical field ignoring other properties of person.

e.g. In main method, we can see only method name at time of calling, we didn't see the behaviour / what is happening inside that method.

2. Encapsulation :
- Binding of data & code together is called as encapsulation
- Implementation of abstraction is called encapsulation.
- It always describe inner behaviour of object
- Function call is ~~encapsulation~~ - abstraction
- Function defination is encapsulation.
  - Unprocess raw material ⇒ data
  - Processed data after operation ⇒ Information

Modularity
dividing programs into small modules for the purpose of simplicity is called modularity

3. Inheritance (hierarchy) IS-A
(Composition) HAS-A
hierarchy is a ranking or ordering of abstractions
Main purpose of hierarchy is re-usability.

HAS-A :
One class HAS-A refference of another class as a property

i.e. In one class, we can take a property of object created in another class.

e.g. We create class Student, we take one property dob which is an object created in class MyDate.
therefore class student HAS-A referance of MyDate class as a property.

See technical book class
See & study whole package study HAS-A

day-4
25th April.

Random class :
Random is an in built class of java which will give the random no from the 0 to the entered number
GetRandomNumber

Inheritance : IS-A relation
IS-A relation is nothing but a parent child relation.
Child IS-A type of parent
child inherits attributes and behaviors from parent !! but child also have its own attributes and behaviours.

Java naming convention:

Parent	Base	Super
child	derived	sub

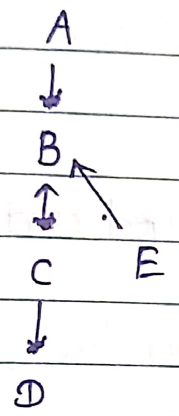
To establish IS-A relation between super & sub class, extends keyword is used.
e.g. Person & employee → Employee IS-A type of parent
∴ when we create Employee object, Person object is created first and then the employee
constructor calling sequence :
Object creation sequence →
Super most class constructor is called first followed by next sub class, followed by current sub class.
A → B → C → D
D is-a C, C is-a B, B is-a A }} Super class A

One class cannot have many super classes. It means one class can have only one super class.

Java does not support multiple inheritance using extends keyword!

e.g. If B is a A then I cannot say B is a Z also (Not allowed)

Advantage of Inheritance : sub class can REUSE the methods ~~of~~ and properties of super class



Study IS-A test

super();

- This keyword calls the no-parameter/default constructor of the super class
- IF written, this line MUST be the first line of the code in CHILD constructor
- IF not written, compiler will add it on its own automatically.

super.show()

This will call the show method of the superclass

super keyword Acts as "this" of a person (superclass)

apsara

Date: __/__/__

Object class :

Object class is the root/super class of all the classes by default.

Root of java object hierarchy ||

We do not write extends object but still object becomes the super class of every class by default

toString method.

toString is an object class method.

it returns a string made up of

Package qualified class name @ hexa address

When we put object in sysout

System.out.print(obj)

internally, obj.toString() is called and the result is displayed.

If we do not write toString in class, it will call toString of super [object] class

but to get readable output, it is recommended that all sub classes should override the method.

Override a Method :

Writing another ~~method~~ implementation in the sub class of the same signature method already present in super class.

i.e. toString is method of object class [defined in object class which is super class of all classes]

in any sub class, if we call toString, it will give toString of object class, but we want o/p in diff way than ~~the~~ toString in object class so we change it in subclass by overriding toString method of super class.

Overloading : Constructor / function overloading
Overriding : method / function overriding

Polymorphism :
Poly - many
Morphs - Forms / implementation of methods

Overloading : One method name, Many forms but the signatures are different.
e.g. Pubic MyDate ( ){..} , public MyDate(int a, int b) {---};
Overriding : One method signature, Many forms signatures are same.
name of function is same but the code in it is different and we have to write it in diff classes.

When we create an object, we can call all its visible super class methods + Its own methods.
study.IS_A.testobj , study.IS_A.Inheritance.

class Alpha fa();
↕
class Beta fb();
↑
Class Theta fc();
↑
class Gama fd();

class Delta fe();

In each object of class, object class method and all of the methods present ~~it~~ in its super class can be called , study.IS_A.Inheritance.

day-5
25th April.

We make objects by,

Alpha obj = new ~~Beta()~~; Alpha();

but, in java we can also make objects by using

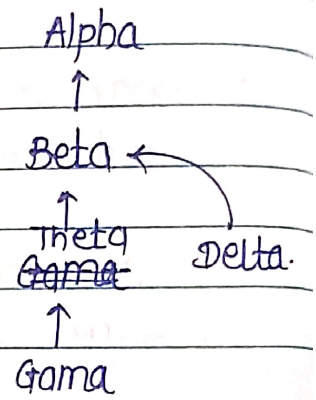
Alpha obj = new Beta();

This will be possible only if Beta is a sub class of Alpha.

The equation is valid if and only if RHS is subclass of LHS

Alpha obj = new Beta(); valid
Beta obj = new Gama(); valid
Alpha obj = new Theta() valid
Theta obj = new Delta() Invalid
Beta obj = new Alpha() Invalid.

LHS. RHS



RHS should be subclass of LHS.

We can call methods also by using these objects.

homework. IS_A. Inheritance.

Alpha obj = new Beta();

Referance obj has two data types

1. The data type used at Compile time — static/ compile time datatype — LHS - Alpha
2. The data type used at Run time — Dynamic/ run time data type — RHS - Beta

Study. IS_A. test Inheritance.

Object class is a superclass of each & every class

Imp # Typecasting

Upcasting : Alpha obj = new Theta();
sub class object is promoted to super class refference.

Down Casting:
Alpha obj ; compile time datatype.
((Theta) obj). Fc();
super class refference is cast to a sub class type.

Equals method of object class.
Super class object implements the equals method as follows:

```
class object
{
    public boolean equals (Object obj)
    {
        if( this == obj)
            return true;
        else
        return false ;
    }
}
```

All classes have two options
1. Reuse equals();
2. Override equals();

default implementation of equals is comparing the addresses of object but if we want to compare content, we need to override equals() method.

apsara

Date:

Keywoted - Abstract

Abstract keyword is used for methods or classes
Not used for properties !!!
If a method is abstract, the method has no implementation
Method with implementation are called as concrete methods.
class that contains at least one abstract method MUST be Abstract class.
We cannot instantiate (create object) of Abstract classes.

day-6
28th April

Interfaces in Java

Interfaces are similar to abstract classes.

Need for abstract classes?

→ Abstract class is the base of hierarchy
classes towards the base of hierarchy are more generalised. classes towards the leaf are more specialised

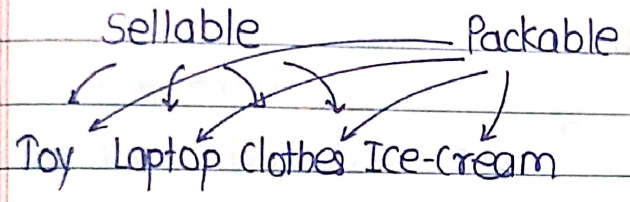
e.g. Shape — Generalized — Super classes

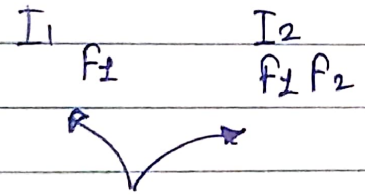
circle Triangle Rectangle — specific. — sub classes.

Advantage of abstract classes: & methods:

- Allow runtime polymorphism so that user of hierarchy can write methods that are low maintenance.
- Even if the hierarchy has more classes, the methods of the user will run without change and accomodate new classes of the hierarchy without any change.

Interfaces: Used for runtime polymorphism in hierarchy
so that user of hierarchy can write methods that will not change even if new classes are added to hierarchy.





$f_1()$ --- f1 is for both

$f_2()$ --- f_2 for I_2

day 7
1st May

Garbage Collector

Implicit memory Management

Implicit : Internally JVM will manage the memory instead of programmer.

JVM Manage which memory?

Heap in the JVM in the RAM

JVM - Java virtual machine ⟹ It is a java process

When we talk about memory management ⇒ JVM is the java process

When we talk about compilers/interpreters ⇒ JVM is interpreter - java command used to run the applicat

Java Process has sections

- Class Area
  All the .class files are loaded here
- Stack Area
  Thread stacks that hold method calls and local variables
- Heap Area
  All the objects are stored in heap area.

All the objects get space on the heap area !!!

This space must be released when objects are no longer used.

If heap gets full, our program will get crashed with a JVM error that occurs at runtime.

java.lang.OutofMemory error

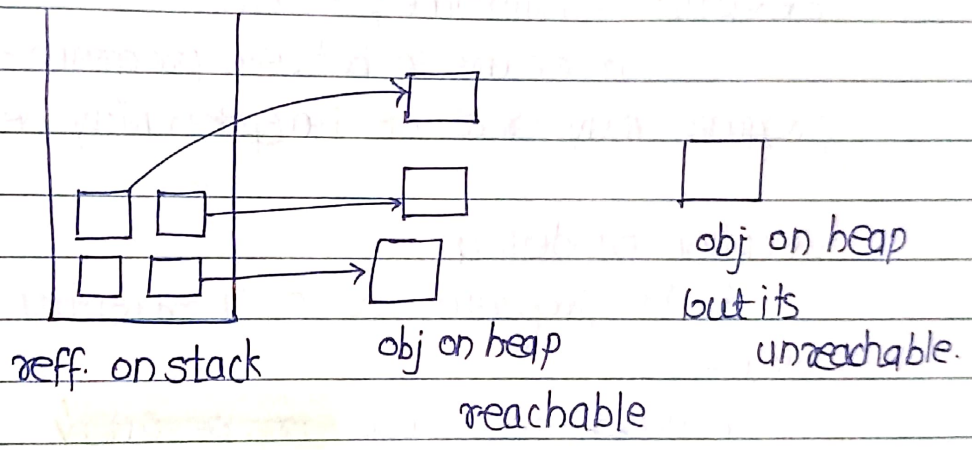
Syntax error → compile time error → class file is not created

JVM error → run time error → it occurs in between program execution & program crash

Exception → run time error → it occurs in between program execution & program may crash or execution may be handled.

Garbage collector

GC is the thread of jvm will run periodically, It will trace the objects for Reachability from stack. It will mark all the unreachable objects and then sweep all the marked objects.



GC calls a method finalize(); on an object ~~when~~
↑ method of object class.

This method is called by ~~GC~~ GC after marking before sweeping

Implicitely GC will be called after some algorithms schedulety

Explicitely GC will be called as system.~~gc~~gc(); to clear memory.

Good Programming Practice for GC to be able to mark and sweep.

As far as possible, keep the scope of reff variable as low or less as possible so that object quickly become unreachable and GC may release them.

Whenever you finished working with a reff, make it null!!

The object it is pointing to may become unreachable and available for GC

Exception handelling

Exception : Run time problems:

It occurs in between program execution and program may crash or Exception may be handled.

Exception handelling:

Avoid program crashing if exception occurs at run time.

To avoid this :- use try & catch

day-8

9rd May

To handle exception, we use try & catch block.

```
try
  {
      problematic code;
  }
catch (exception)
  {
      hadle;
  }
```

When we talk about the exceptions/errors, there is a library for exceptions & errors is present (API)
An API (Library for exceptions/errors)
Many readymade classes are present.

Under Object class, in default package (java.lang), we have class Throwable: It has two subclasses
Error & Exceptions
Each error or exceptions are classes in java. i.e. All runtime exceptions & errors are classes under throwable class.

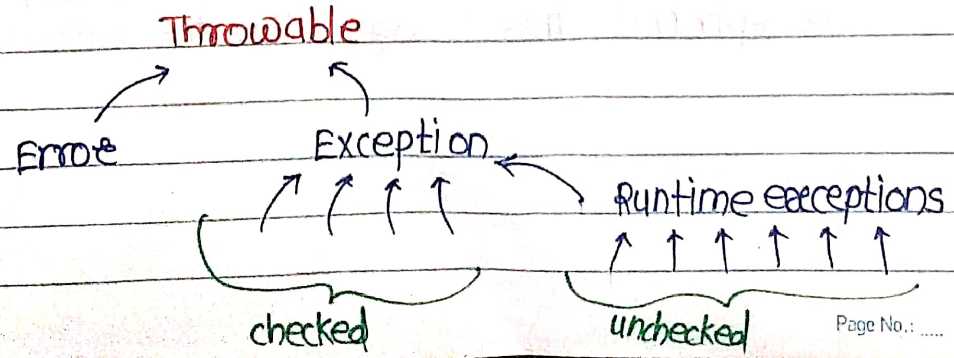
Throwable

Error	Exceptions.
All error comes under this error sub class of superclass throwable	All exceptions comes under this Exceptions sub class of super class throwable.

Exceptions are of two types:
a) checked exepections
b) Unchecked exepections

checked exceptions
All the subclasses of Exception class except Runtime Expection class sub hierarchy

Unchecked exceptions
All the subclasses of Runtime exception.



3 blocks in exception handlling

1. try - One try can have 0 or more catch blocks
   0 or 1 Finally block
   Either catch or finally can be present
2. Catch - Will catch one or more exceptions and handle it, if the control reaches to catch block then exception can be handled.
3. finally - One try can have max one finally block.

(Finally block will always execute → The code that must run in any situation must be written in finally block.

One try can have many catch blocks or
One catch ~~cat~~ can catch many exceptions.
When we have multiple catches -->
The super class catch must be in the end and the subclass catches must be above, otherwise code does not compile → Unreachable syntax error.

Keyword Throw & throws

throw : It is written inside a method
throw object of any exception class.
throw will generate explicitly an exception.

throws : It is written in signature of the method.
throws name of exception class or classes
throws declare that the given method may throw an exception !! This is useful for the caller of the method.