Core Java. 11/2/2022

Java

1) Java was developed by James Gosling, Patrick Naughton & Mike Sheridan

2) It was initiated in 1991 & was called as Green.

3) ~~[illegible]~~ Finally in 1995 Java ~~[illegible]~~ 1.0 was released.

OOPs (Object Oriented Programming)

1) The first property of oops Encapsulation
   → Encapsulation in java is process of wrapping code and data together into a single unit
   Ex: Capsule.

2) Next Principal is Inheritance
   → Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of a parent object.

3) Polymorphism
   → Polymorphism in java is the ability of an object to take many forms.
   i) compile-time polymorphism - Method overloading
   ii) Run-time Polymorphism. - Method Overridding

4) Everything must be written within the class
5) Everything must be created in terms of classes and object.
6) Every action must be performed by method invocation.

Java is ~~does~~ not ~~[illegible]~~ 100% OOPs because it support primitive datatypes.

Java features

1) Simple
2) Pure OOP.
3) Platform independent.
4) Portable
5) Secure
6) Robust
7) Dynamic
8) Multi-threded.
9) Exception-handling
10) Distributed. (RMI)
11) Both compiled and interpreted [Ex.→ Java, typescript]

Java Edition

Java SE
Java EE
Java ME

. class file is byte code JVM is understand that.

class, object, Constructor.

Variable (It is a container which contains value)

i) Instance
ii) Static
iii) local.

i) Instance: It should be directly declare inside the class.
It should ~~be~~ not be declare any block ~~o~~ or class
int temp = 12;

ii) Static :- It should be declare inside the class using static keyword.

static int temp2 = 12;

iii) Local: local variable are declare inside any block

```
public static void main( --- )
{
  int temp3 = 10;
}
```

- We using instance variable by using object.
- static variable access two ways
  i) By using object of the class.
  ii) By using class name ~~as a ode~~

Instance & static variable is initialize by java compiler.
But local variable is not initialize by java compiler.

Q. why static variables can be accessed without creating objects with just ~~creating object~~ class name?
⇒ The static variable allocate memory while loading class ~~and~~ itself in method area.

Q. Why java doest support constructor overriding?
⇒ • java will consider ~~a~~ that as a method and show error ~~[illegible]~~ (return type).

Data type

1> Primitive (1. byte, 2. char, 3. int, 4. float, 5. long, 6. Double, 7. boolean, 8. short)

2> Non- Primitive.

classes, Array, enums, interface. etc.

boolean →	1 bit [range → -128 to 127]
char →	2 byte
byte →	1 byte
short →	2 byte
int →	4 "
long →	8 "
float →	4 "
double →	8 "
class →	8 byte.

Java have 4 conditional statement.

1> if statement
2> if - else "
3> while "
4> switch "

Encapsulation (In a concept of hiding data)

[illegible] In encapsulation we use getId() & setId() method.

Abstraction

Using i) Abstract class, ii) Interfaces.

• Abstraction is a process of hiding of internal implementation and show only the functionality.

We can achieve abstraction in two ways

i) Abstract class.

ii) With Interfaces.

✓ With abstract class we cannot achieve 100% abstraction because abstract class allows both abstract class & abstract method, with interfaces we can achieve 100% abstraction. because it allows only abstract method

✓ If your class have a method, method body then the class is called abstract class and method will be taken as abstract method. we cannot create object of Abstract class.

✓ [illegible] If your class is extending abstract class, and if your class having abstract method, either your class is declare as abstract class or you ~~will be declare~~ have to implement the abstract ~~class~~ method in your class.

Inheritance

1. Single Inheritance
2. Multilevel "
3. Multiple ". [Java not support]
4. Hierarchical Inheritance

Why Java does not support Multiple Inheritance?

→ It creats diamond problems.

It is used to inherit the property of an parent class to a child class

ex:
```
class A {
        int i=10;
      }
   class B extends A {
        system.out.println(i);
      }.
```

Single Inheritance

This is simple one inheriting property from one class to other class.

Multilevel.

One super class for each sub class.

```
  class A {
          int i = 10;
  }
 class B extends A
       {
         sysO (i);
  }
 class C extends B {}.
```

Multiple

Multiple super class for each single subclasses.

```
   class A {
            int i = 10;
          }
      class B ~~extends~~ {}
         class C extends A,B {}
```

Hierarchical.

More than one class is inherited from a single parent class.

```
class A {
      int i = 10;
 }
class B extends A {

           }
 class C extends A {

           }.
```

Interface	Abstract class
i) We ~~don't~~ can't have constructor in interfaces.	i) We can have constructor in Abstract class.
ii) We ~~will~~ can have only Abstract method.	ii) We can have both the method i.e normal & Abstract.
iii) We can have by default public, static & final variable	iii) We can have any type of variable in Abstract class.
iv) We can achieve 100% abstraction.	iv) We cannot ~~[illegible]~~ achieve 100% abstraction.
v) We can achieve multiple inheritance.	v) We cannot achieve multiple inheritance.

Q. To create object we need constructor why we cannot create object of abstract class?

→ Because there might be chances that ~~they migh~~ Abstract class may not ~~[illegible]~~ have abstract method.

1) Class can extends class.

2) class can implements interface

3) ~~class~~ Interface can ~~implements~~ extends interface.

Polymorphism

i) Compile-time polymorphism (overloading)

ii) Run-time polymorphism

Packages ~~[illegible]~~

1) Why we need packages?

=> i) code reusability

ii) To avoid naming collision

2 packages { Java
Javas

Syntax:

```
javac -d . fileName
```

Access specifier.

i) Private, 2) Default, 3) Protected, 4) Public

1) Private: We can access the private variable throughout the class.

2) Default: We can ~~[illegible]~~ access throughout the class and througthout the same package.

3) Protected: We can access throughout the class, ~~[illegible]~~ throughout the package, ~~sub-package~~ class of sub-package with extends the class. ~~[illegible]~~

4) Public: We can access from any package.

Q. Why we have main method public?
→ Because we can able to access in anywhere

Marker Interface (The interface which does not contain any thing or empty)
1) ~~[illegible]~~ cloneable interface
2) serializable interface

Exception.

1. Compile Time Exception/ checked Exception : all io & sql Exception are checked Exception
2. RunTime Exception/Unchecked Exception:
   Arithmetic Exception, IndexOutBound, security, class CastException.

✓ Exceptions are unwanted & unexpected event occurs that disturb the normal flow of execution is called exception.
   Ex: Arithmetic Ex, null pointer ex, Array Index Out of Bound Ex.

✓ Main objective of Exception Handling is to greatful termination of the program

✓ Handling Exception does not mean repairing Exception it is to provide alternative to continue rest of the program normally.
   Ex: if you trying to access online file in case it is not available then file not found exception will throw by jvm and terminate the program, to avoid abnormal termination of the program we need to provide local file to continue rest of the execution of the program normal.

✓ Two ways to handle checked exception,
   i) Throws keyword
   ii) try-catch block.

try Block

1) Risky code means the code which raise exception should be write in try block

2) try Block is used handle both check and unchecked exception.

Catch Block.

1) In this block the Exception handling code will written.

2) If we did not handle the Exception then default Exception handler will handle the Exception and terminate the execution.

finally Block.

1) finally Block will always execute regarding of raising exception.

Diff final, finally & finalize()

Throw keyword

Throws keyword

Throws is a keyword used to handle checked exception
ex: io Exception.

Throws keyword does not handle unchecked exception for that we need to use try catch block to handle run time exception.

Using throws keyword ~~when run time~~ (always use for compile time exception)

Exception wierarchy

Throwable class act as root class for all java Exception.

Throwable class define two more child classes.

1. Exception 2. Error.

Exception are mostly caused by our program and these can be recoverable.

Error are mostly caused by lack of system resources and cannot be recoverable.

Checked Exception vs Unchecked Exception

Checked Exception	Unchecked Exception
1> This exception checked by compiler.	1> compiler will not check this Exception.
2> Throws key word and try catch block is used to handle this exception. Ex: io exception.	2> Try catch block are used to

Handling all type of Exception:

1> We can handle all type of exception in catch block.

2> Write number of catch block as number of Exception you want to handle.

3> Order is important for that you can give catch (Exception e) in last

4> You can write only one catch block to handle custom Exception.

1) User defined Exception is called as custom exception.
Ex: invalid password, or user name or insufficient fund etc.

2) We can create both checked and unchecked Exception

3)

4) If you want create run time Exception then you need to extends your class with Runtime Exception
Syntax:

class ClassName extends RuntimeException

Q- Can we write try without catch?
Ans:- No try always comes with catch block.
Q. can we write catch without try?
Ans:- No catch always followed by try block.