4-4-2024

Page No.

Date / /

method < concrete
abstract

Interface contains constant variable & abstract meth
does not contain concrete method.

Abstract class: (don't have a definition)
Abstract class may contain concrete method
concrete method means it contains class definitio

Abstract class should contain ~~at~~least one abstra
method

```
Interface Voter {
        void castVote();
}
interface EC {
        void checkValidityOfUser();
}
abstract class ECIndia implements Voter, EC {
        void castVote(){
        }
        void checkValidityOfVoter(){
        }
}
```

4-4-2024
Page No.
Date / /

method < concrete
abstract

Interface contain constant variable & abstract method
does not contain concrete method.

Abstract class: (don't have a definition)
Abstract class may contain concrete method
concrete method means it contains class definition

Abstract class should contain atleast one abstract
method

```
Interface Voter {
        void castVote();
}
interface EC {
        void checkVailidityOfUser();
}
abstract class ECIndia implements Voter, EC {
        void castVote(){
        }
        void checkVailidityOfVoter(){
        }
}
```

10-4-2024

Page No.

Date / /

~~192-II~~

error specific question
Exception handling
interface, abstract class
file handling

Throw exception

If their is a relation betn the two things like ECIndia & ECMaharashtra then you need to go with classes.

public is a weakest access specifier.
private is the weakest

gedit
vi — terminal to exit from this command press W.

abstract class Sample

error: missing method body or declare abstract.

abstract class Sample {
    void funSampleAbstract();
}

you need use abstract keyword.

abstract void funSampleAbstract();

then it will compile.

error: interface abstract method cannot have body

work with constant variable

Page No.
Date / /

```
abstract class Sample {
        abstract void funSampleAbtract();
        void fun() {

        }
}
```

error: sample is not abstract and does not override abstract method funSampleAbstract() in sample.

```
class sample {
        -11-
}
```

@ You cannot create the object with abstract class.
Abstract class can be superclass.

Abstract class cannot hold reference ~~ver~~ or instance.

Page No.
Date / /

Inner class.

If we use private access specifier before a class, the class is not available to the java compiler or JVM. But, private specifier is allowed before an inner class & thus it is useful to provide security for the entire inner class.

```
abstract class Sample {
        abstract void funSampleAbstract();
    }
class Demo extends Sample {
        void funSampleAbstract() {
                S.o.p("Inside Demo");
        }
class Relate {
        P.S.V.M(String args[]) {
                Demo D1 = new Demo();
                D1.funSampleAbstract();
    }
```

Abstract method is written when the same method has to perform different tasks depending on the object calling it.

Page No.
Date / /

In interface variables are static & final & methods are public & abstract.

Page No.
Date / /

Checked Exception :- The classes which are directly inherited from the class Exception represents checked Exception. For these exceptions the compiler causes unreported exception (error) at the time of compile to forcefully implement the exception handling mechanism.

Unchecked Exception: The classes which are inherited from the class RuntimeException and from class Error represents unchecked exception.
For these exceptions the compiler not cause any compile time error.

* Abstract class.

```
abstract class Sample
 {
    void fun();          // will not work missing method body or
 }

abstract class Sample{
    void fun(){          // will work.
    }
 }

abstract class Sample
    abstract void fun();
    void funSample(){ }
}
```

We cannot create the object of abstract class. We can define abstract class reference & create subclass object.

Page No.
Date / /

Interface

→ It is the collection of final data members & abstract method
→ Sometimes, it only contains abstract methods
→ Methods in interfaces is by default by public and abstract, so we don't need to use prototype.
→ The data members in the interface is by default public, final and static (i.e implicitly)

The interface is used to acheive multiple inheritance & polymorphism.

Syntax:-

```
access_specifier interface i-name
{
    final_member;
    ...
    abstract method;
    ..
}
```

Page No.
Date / /

1) public java.lang.String getMessage();
→ Returns the detail message string of this throwable instance. It's the message passed to the constructor or null if no message was passed.

→ Index 0 out of bounds for length 0.

2) getLocalizedMessage()
→ Index 0 out of bounds for length 0.

3) toString:
returns a short description of this throwable. The result is concatenation of the name of the class of this object.
java.lang.ArrayIndex -11 — : Index 0 out of bou; -11-

4) e.equals(e)
⇒ true.

5) getCause()
returns the cause of the exception or 'null' if the cause is unknown.

6) fillInStackTrace();
This method fills in the execution stack Trace. Mainly used by the JVM or by methods that need to capture or preserve a particular stackTrace.
-11-

Page No.
Date / /

7) getStackTrace()
returns an array of stack trace elements representing the stack trace of this throwable.
[L java.lang.StackTraceElement;@251a69d7

8) initCause()
initializes the cause of this throwable to the specified value. It can only be called once. It returns a reference to this throwable instance.

9) getSuppressed()
returns an array containing all of the exceptions that were suppressed, typically by the try-with-resource statement.

10) printStackTrace()
print the stack trace of this throwable to the standard error Stream.

```
void modify (int x){
  this.x = x; // store local variable in present class instance var
}
```

2-4-2024

* Inner class.

Exception → checked
Unchecked.

If we are getting exception at the time of compilation then we can check the exception.

Compile time exception.

Runtime exception → Unchecked (arrayIndexoutof bound)
→ Error (Stackoverflow)

Illegal operation: Divide by zero.

In the recursive call we keep on creating objects then the memory of JVM will be full & then the stackoverflow error occurs & it is not intended (in the program)

* Command-line arguments (explore this concept)
Ques ⇒ java ffirst java ffirst
⇒ java.

java ffirst 123.456 567
⇒ 123.456

Page No.
Date / /

Parameter :- Whatever you are providing in function signature

public static void main (String ... args) {

}

The above is the valid syntax

(String ...args) ← It will give an error.

Unchecked

public static void main (String ... args) {

S.O.P (args[0]);

}

It will compile but.

At the time of execution

Exception in thread "main" java.lang.ArrayIndexOutOfBoundException

Index 0 out of bounds for length 0.

class FFirst {

P. S. V. M (String ... args) {

try {

S.O.P (args[0]);

}

Catch (Exception e) {

S.O.P ("Exception at line Number 5");

}

}

}

3-4-2024
Page No.
Date / /

getMessage()
getLocalizedMessage()
toString()
printStackTrace()

error comes at ~~the time~~ runtime.
or at the time of execution.

Parent class can use child class object

Exception class ~~ob~~ can make a call to method of Throwable class because Throwable is a parent class & exception class inherit the properties of throwable class And another point if it is private then we cannot access it.

error: 'try' without 'catch' 'finally' or resource declaration.

```
P.S.V.M (String ...args) {
    try {
        S.O.P (args[0]);
    }
    finally {
        S.O.P ("Exception at line");
    }
}
```

Page No.
Date / /

error: 'catch' without 'try'

```
P.S.V.M(String ...args){
    try{
        S.O.P(args[0]);
    }
    catch{
        S.O.P("Inside catch");
    }
    finally{
        S.O.P("Inside finally");
    }
}
```

O/P ⇒ Inside catch
Inside finally

~~java Example 0~~

java classname 0x8

O/P ⇒ 0x8
Inside finally.

checked exception ⇒ error: unreported exception InterruptException; must be caught or declared to be thrown.

Page No.
Date / /

```
try {
    Thread.sleep(1000)
    s.o.p (args[0]);
    Thred.sleep(1000);
  }
catch (ArrayIndexOutofBoundsE e) {
    S.o.P ("Inside ArrayIndex -11- catch");
  }
catch (InteeruptException e) {
    S.o.P("Inside Inte -11-  catch");
  }
finally {
    S.o.P("Inside finally");
  }
}
```

If i am teying to peeform some activity but i fail to peeform that time exception occue.

```
Catch (Exception e) {
    S.o.P("Inside Exception catch");
  }
```

Add this catch block at first & last & see the result.

```
P.S.V.M (string ..args) throws InteeruptedExcep
```

Page No.
Date / /

Compile time error: This error occur during the compilation phase of a program.
ex: Missing semicolons, undeclared variables.

Runtime error: This error occur while a program is running.
cause: division by zero, stack overflow.
Ex: Accessing an array element beyond it's bounds.

logical error: (semantic error) when the program compiles and runs without error message, but does not produce the expected output due to flaw in the algorithm.

ex: Incorrect condition, wing wrong formula.

* Exception are runtime error.

Runtime ⇒ write main method in java without it's parameter String args[]
Runtime error are detected by JVM, only at runtime.

* The exception that are checked at compilation time by the java compiler are called checked exception.
* while the exception that are checked by JVM are called unchecked exceptions.

Unchecked exceptions and errors are considered as unrecoverable.
In case of other exception (checked) programmer should either handle them or throw them without handling them.

Page No.
Date / /

```
Try {
}
Catch(ArrayIndexOutOfBoundException){
    ....
}
Catch(InterruptedException ex){
...
}
Catch (Exception ex){
...
}
```

Output:- Work Properly.

```
try{
...
}
Catch(Exception ex){
...
}
Catch(ArrayIndexOutOfBoundException ex){
...
}
Catch (InterruptedException ex){
...
}
```

Output : Error.

Page No.
Date / /

```
class Ex
{
P.S. void m (String args[]) throws InterruptedException {
    try {
        Thread.sleep(1000);
        S.O.P("Try 1");
    }
    catch(Array ++ ) {...}
    catch(Exception ex) {...}
} }
```

⇒ Work Properly.

Exception classes Hierarchy.

class Object
↓
class Throwable

class Exception
- class IOException
- class SQLException
- class InterruptedException
- class RuntimeException
  - ↳ class ArithmeticException
  - ↳ class NegativeArraySizeException
  - ↳ class NullPointerException
  - ↳ class IndexOutOfBoundException

class Error
- ↳ class VirtualMachine Error
- ↳ StackOverflowError
- ↳ OutOfMemoryError

Page No.
Date / /

command line arguments.

```
class ffirst
{
    public static void main(String args...) {
        S.O.P("Inside main");
} }
```

error: ',' ';' ')' or '[' expected.

```
class ffirst{
        P.S.V.M (String ... args){
            S.O.P(args[0]);
    } }
```

n$:- javac
it will compile successfully
n$: java ffirst ~~argg~~
~~ns~~

arg
n$: java ffirst (means without any argument
Exception: ArrayIndexOutOfBoundsException:
Index 0 out of bounds for length 0.

```
* class ffirst {
        P.S.V.M (String ...args){
          try {
              S.O.P(args[0]);
          }
          finally {
              S.O.P("Exception at line number");
} } }
```

Page No.
Date / /

You can write a try block without a catch block, but if you do so, you must include either a 'finally' block or both 'catch' and 'finally' blocks.

Error: Catch without try

This error occurs when you have a 'catch' block without a preceding 'try' block. The purpose of the 'catch' block is to handle exceptions that are thrown within the corresponding 'try' block.

```
ex: class fFirst {
        p.s.v.m(String ...args) {
            s.o.p(args[0]);
        catch(Exception e) {
            s.o.p("Exception at line
                number");
    } } }
```

Checked exception:

Unreported exception InterruptException; must be caught or declared to be thrown

occurs when you call a method that declares that it throws a checked exception, but you don't handle or declare that exception properly in your code.

Page No.
Date / /

In Java when catching exceptions, the order of catch blocks matters. More specific exception types should be caught before more general ones.

the arrangement should be like

```
catch (ArrayIndexOutOfBon— )
catch (InterruptedException)
Catch (Exception e).
```