**JAVA**

**Jshell-type in Cmdr. Jshell then enters in jshell for exit /exit**

* **J shell referes to java shell tool**
* **We can write,execute and test java code snippets in a single line within this shell**

**Exercise 1**

**Write an expression to calculate min in day and second in day**

**->1hr=60min 24hr=60\*24=1440min  
-> 1hr=60min 1min=60sec 360sec 1hr 360\*24=8640**

**A data sheet with text

Description automatically generated with medium confidence**

**A black screen with white text

Description automatically generated**

**Java Platform Indepedent**

**A diagram of a computer program

Description automatically generated**

**Class- class is blueprint of object class contain methods function and all**

**Class country{ }**

**Object-object is physical representation of class**

**Class name p1=new class name();**

**Using new key word we make object**

**Coding Exercise: Is Leap Year**

return (year != 0) && ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0));

Perfect Number calculate

**Task 1**:

* Operators
* Variables and Data Types(done)
* Control Flow (if-else, switch-case, loops)
* Exception Handling (try-catch-finally
* Variables and Data Types(done)

In variable we can store data

Datatypes-

|  |  |
| --- | --- |
| Primitive/inbuilt | Non Primitive/userdefine datatypes |
| Int, float, char, long, Boolean (8types) | Array,String,List |
| Byte,short, |  |

A screenshot of a computer

Description automatically generated

**Type casting-**

Convert one data type to another data type

Two types of

1)implicit type casting

Automatically by jvm

Int to float,double

2)Explicit

Forcefully conversion

Double val=23.23;

Int val2=val;// error loss chances only 23 and .23 loss

Int val2=int(val1);// forcefull conversion

**A screenshot of a computer

Description automatically generatedOperators**

**Ternary operator**

**A+b ? true: false**

**Control Flow (if-else, switch-case, loops)**

**1)If statement-**Condition based execution

**Ladder if**- if(condition 1){

syso}else if(condition 2){

}

**2)switch Statement-cases based**

**Value=0**

**Switch(value){**

**Case 0;**

**Syso(“bro”);**

**Break;**

**3)LOOPS=repeating things statement**

**I]for loop-**for(initial;termination;incr/decr**)**

**II]while(condition){its entry control loop**

**Statement**

**}**

**III] do while=its exit control loop**

Do{statement}

While(condition);

**4) For-Each loops=**for traverse array

For(int x:num)

------------------------------------------------------------------------------------------------------------------------------

**Object-Oriented Programming (OOP)**:

* Classes and Objects
* Inheritance
* Polymorphism
* Encapsulation
* Abstraction

Program-----🡪 style(methodology)----🡪 -----procedure Programming

Functional-Programing

**Class and Object OOPS**

Class=its logical Entity 🡪he have properties(data) and behavior(methods)=>when you create objects of that class then your class will be an real entity(data and method ) => with same class we can create multiple objects

Class Students{

//data: data members: instance variables

Int id;

String name;

String City;

//this is behavior :method: function

Public void(){

Syso(name +”name of stud”);

}

* **Constructor**

Constructor name is same as of class name

Contructor is used to initialize the data of the object

**A screenshot of a computer code

Description automatically generated**

No need to initialize the data like this using constructor we can initialize the data

When we create object constructor automatically call

1. Name==Class name
2. Does not return any value ,not even void
3. We can add parameters(parameterized constructor)
4. No parameter (non para)

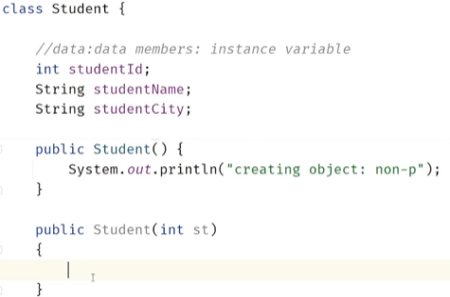
**Types =>**

* **A screenshot of a computer

  Description automatically generatedParameterized Constructor -we can pass parameter**
* **Non-Parameterized Constructor-don’t have parameter**
* **Default constructor-automatically called by jvm**

**Overloading-**

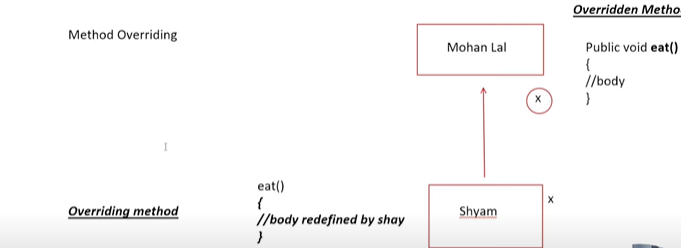
* **Constructor overloading-We can create multiple constructor In same class is called constructor overloading**
* **When we create multiple methods called method overloading**

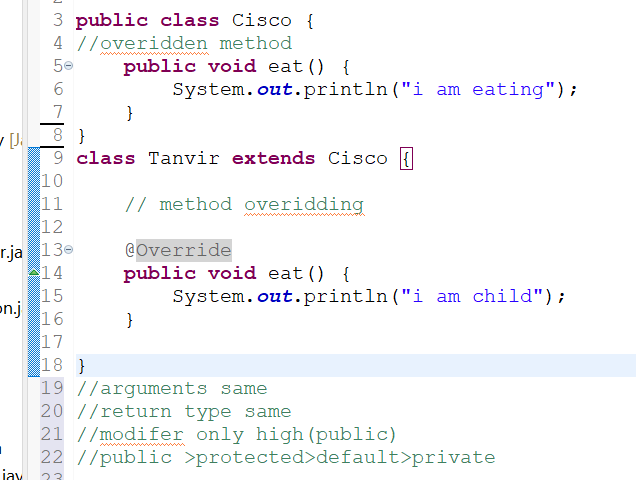
****

1. **No. of arguments different**
2. **Types of arguments different**
3. **Order of arguments may be different**

**Q) how to call parameter constructor**

**Student s1=new Student(23); //passs int value**

**Overriding –** in parent class and in child class having same method with same name with same parameter is called overriding

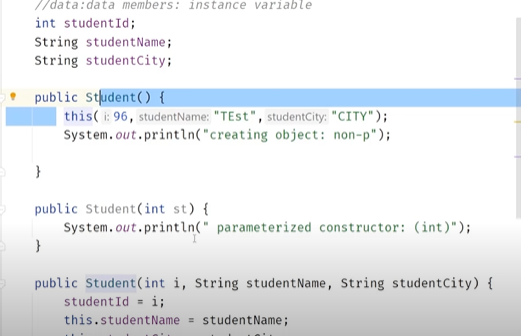


**Keywords🡪**

1. **This keyword**

**refers to current class object**

**this()🡪For calling current class constructor from inside different constructor of sam e class**

****

class Test(){

Public void Show(){

This.a=a;

}

}

Test t= new Test()

t.show();

**NOTE-We can call the constructor in other constructor using arguments**

E.g. . constructor (){

This (96,23) ;/// calling sec constructor

This (96,23,23)// calling 3rd constructor

}

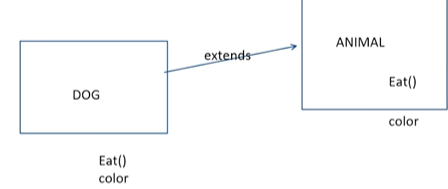
Constructor(int a, int b){,}

Constructor(int c ,Int d, int f){}

1. **Super keyword🡪**Calling superclass constructor in parent class.

**Inheritance🡪**

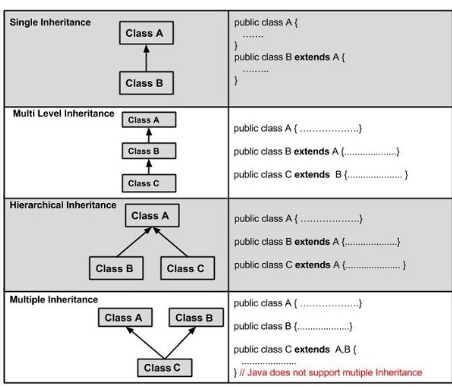
**One class inherits the feature of other class**



Animal**🡪**Parent class, Super class, base class

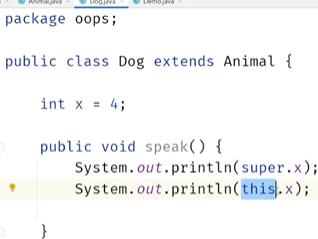
Dog 🡪child class, sub class, derived class

Class Dog extends Animal{}

**Types Of Inheritance**-

**Super Keyword-**

Super keyword used to call immediate super class objects

****

Note- in animal class x=12 is declare but in Dog class also X=4 is declare so,

If we want to call parent class object in child class we used super.x

This key is used for current class object.

Super() method used for call super class constructor //super, this ,must be first line

**Java Inner Classes**

In Java, it is also possible to nest classes (a class within a class). The purpose of nested classes is to group classes that belong together, which makes your code more readable and maintainable.

To access the inner class, create an object of the outer class, and then create an object of the inner class:]

A computer code with black text

Description automatically generatedIf inner class is private then we cant call it

**private** class InnerClass {int y = 5; }

**Encapsulation🡪**

* binding or wrapping all data or code in single entity.
* E g. watch getter and setter method is used
* Data hiding and security

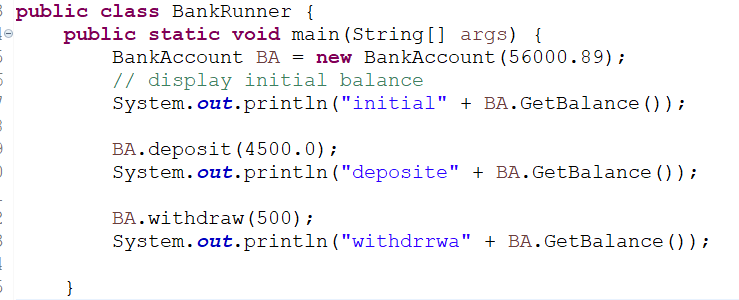
Class test

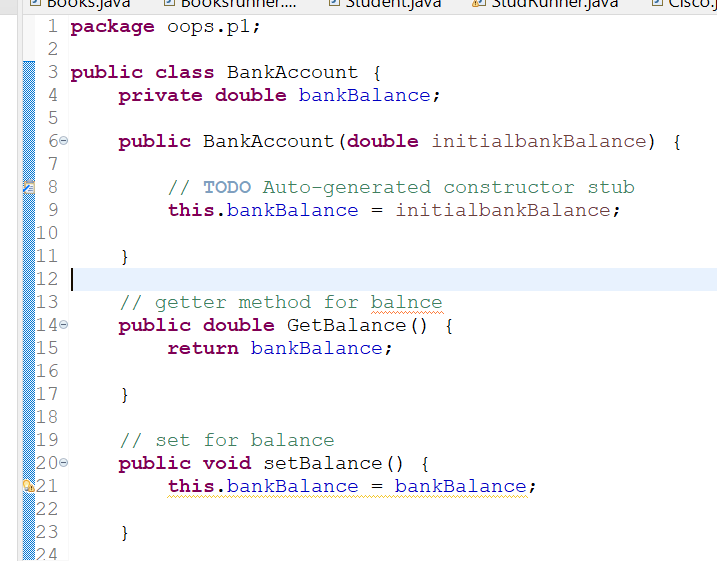
{

private Int a=12; //always private outside not access

Public void show (){

**}**

****getter and setter method

**}**

**Program Encapsulation:🡪**

**Abstraction🡪**

Abstraction means presenting essential features or behaviors of a system, while hiding unnecessary details and representations

* e.g. Car, ATM ……. in car we know essential features or behaviors like how to learn operate, while hiding the unnecessary details like internal design, stroke engine
* Abstract class and interface

Note-if u want 100% Abstract=>go with interface

Note-in between 0% to 100% abstraction =>Abstract Classes

Abstract Method: if method don’t have any body or 100% incomplete method called abstract method

E.g. public void (){

Tanvir Sayyad //this is complete method

}

Abstract public void cisco(); //this is abstract method

Rules:🡪

* + 1. we can put abstract as well as non abstract method in class
    2. if in class have one abstract method then our main class will be abstract
    3. abstract class cannot create object
    4. how to use without object🡪using implementation 🡪child class

A screenshot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generated

This is 50% Abstraction

* For 100% abstraction using interface
* In class every variable are public static final
* Public static final int x=20;
* If you create normal method this will be abstract automatically
* A screenshot of a computer code

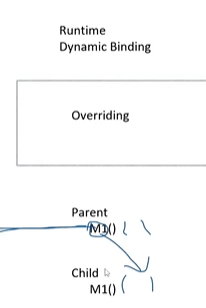
  Description automatically generatedIn this Implements keyword used

**Q} Abstract vs interfaces difference🡪**

**Polymorphism🡪**

* Poly(many)-morphs(forms)
* Runtime polymorphism or runtime binding
* Ability of object to behave differently
* Compile time polymorphism or static bA white rectangular object with black text

  Description automatically generatedinding

****

**1)Compile time polymorphism-> we achieved using Overloading**

Object is bound their functionality at the compile time

It means multiple method with same name name but different in number of parameters and types of parameters

**Eg.** Test () if we know this body will execute at the compile time that’s called compile time polymorphism

**Real time Scenario->ATM machine**

**In an ATM machine we can withdraw money different way**

**-using card**

**-using UPI**

**Programing->**

**Class ATM{**

**Public void withdraw(int pin,int pass){**

**Syso(“using UPI”);**

**}**

**Public class withdraw(String AC,String IFSC){**

**Syso**(“using bank transfeer”)

**}**

**}**

**Class main(){**

**Public static void main(String[] args){**

**ATM AM=new ATM();**

**AM.withdraw(2112,9292)**

**}**

**}**

2)Runtime polymorphism->we achieved using overriding

Just reverse of compile time

Object is bound with their functionality at the run time

Overriding-> overriding means in parent class and in subclass have same method with same paramerters

Example->different types of vehicle start differently like car and bike but they both have start method

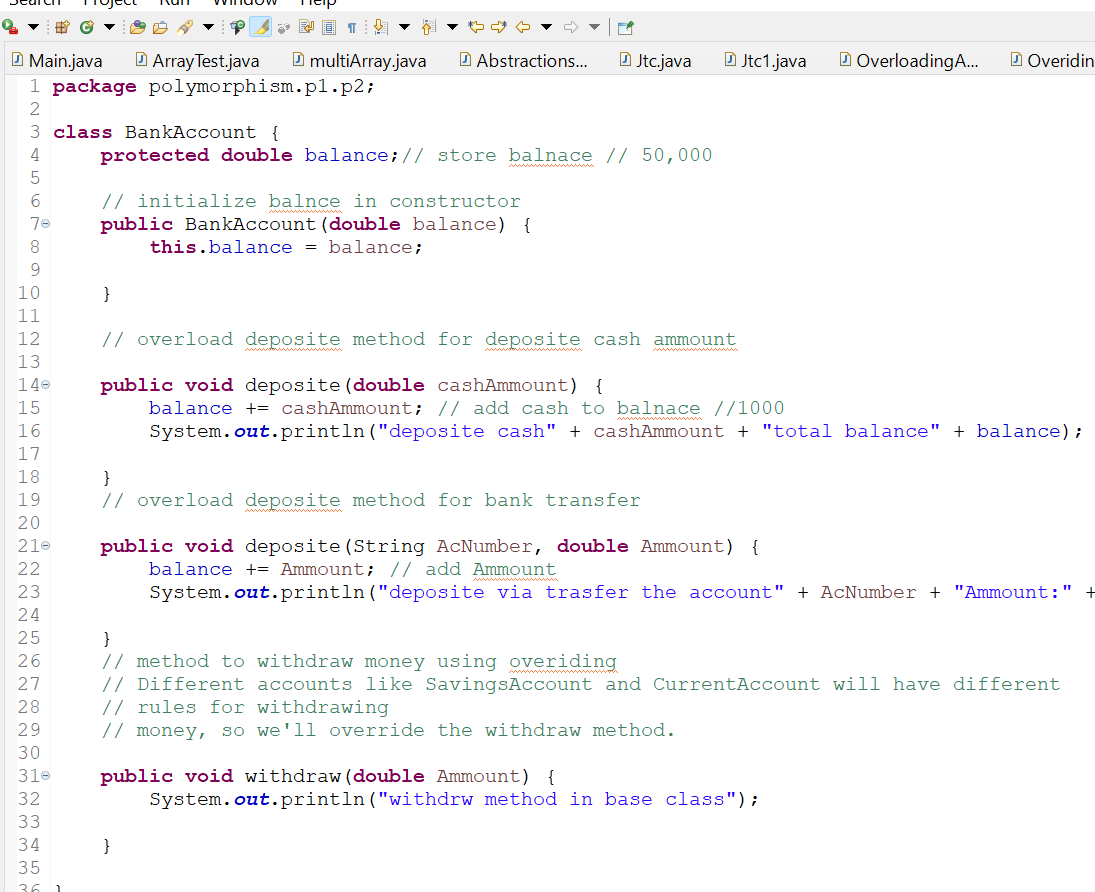
A screenshot of a computer program

Description automatically generated

Scenario: Bank Account with Deposit and Withdraw

 Method **Overloading**: The bank allows deposits through cash or transfer, so we'll overload the deposit method.

 Method **Overriding**: Different accounts like SavingsAccount and CurrentAccount will have different rules for withdrawing money, so we'll override the withdraw method.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

---------------------------------------------------**New**-------------------------------------------------------------------------------------------------------------------

**Packages in java ->**

Scenario-when we click photo we want store photo in other devices while phone memory full that time we can make photo as like pick from college photo, visit photo like this so package doing same similar

Package –

1.naming duplication avoid

2. we can save multiple classes with separately

3.security also increase (by encapsulating classes)

**How to use class in different class**

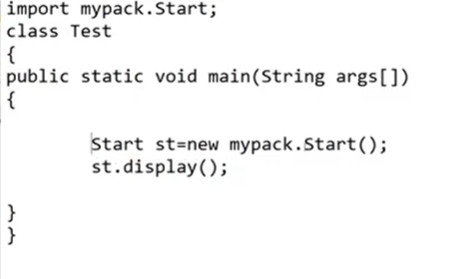
Class which is in one package use in another class

1.using Import keyword for importing the class in our class

-we can use 3 way -Using fully qualified class name ()

-Using single class (import java.util.Scanner)

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated-importing all classes of package (import java.util,\*)

--------------------------------------------------------**NEW**-----------------------------------------------------------------------------------------------------------------

**Exception Handling ,try Catch, finally, throw, throws**

**Exception definition: An exception is an abnormal condition that occurs during the runtime of a program, disruption of normal flow**

Java program flow :

Source code (.java) → Compiler (.class) → Interpreter (JVM)-Output

* First source code(.java) compiled into byte code(.class) by the java compiler
* This byte code(.class) executed by the JVM which interprets the bytecode line by line and produce output.

**Major reasons why exception Occurs**

* Invalid user input
* Device failure
* Loss of network connection
* Code errors
* Out of bound
* Null reference
* Type mismatch
* Database errors
* Arithmetic errors

Types Of Errors:

1. **Compile-Time Errors**:

* This occurs during compilation of the source code usually syntax issue such as (missing semicolon, incorrect spelling, or incompatible data types, Missing braces )
* These error are caught by the compiler before the program is run

1. **Runtime Error(Exception):**

* Exception occurs while the program is running stage
* The error detect at runtime by JVM, after the interpreter starts processing the bytecode
* Example-when we try to divide by zero (Arithmetic exception),accessing an invalid array index

Q difference between Error and Exceptions

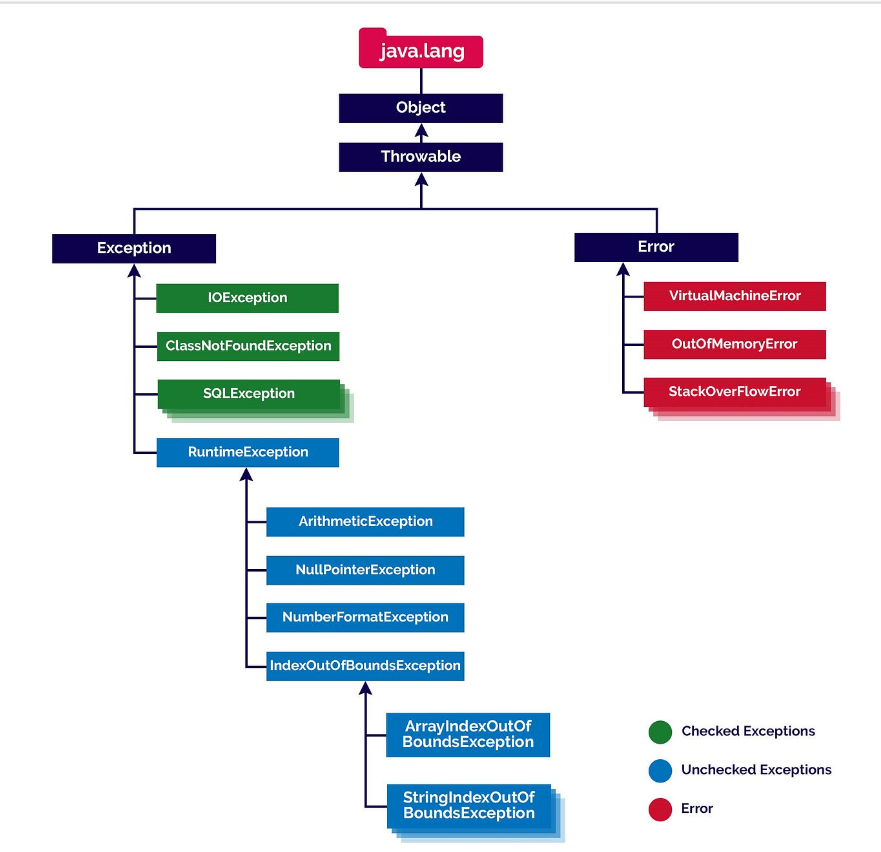
* Error: An Error indicates a serious problem with the JVM or system resources usually beyond application control(eg. outOfMemoryError)
* Exception: Exception indicates conditions that a reasonable application might try to catch

Q. Why handle Exceptions:

* To ensure that the program does not terminate abnormally at mid instead we want program to handle the error gracefully and terminate normally

**Java Keywords for Exception Handling**:

* try: Defines a block of code inside try might throw an exception
* catch: Catches and handled the exception thrown in try block
* finally: A block that is always executed After try and catch ,whether an exception occur or not
* throw: Used to explicitly throw an exception
* throws: The use of throws allows the method to pass the responsibility of handling the exception to its caller, adhering to the definition of throws

**Exception Hierarchy**

Types of exceptions-

1.built in Exception

2.UserDefined Exception

**-Build-in exception**

1. Checked exception /compile time exception

Checked exception are called compile time Exception

* IOException is also called as a checked exception they checked by compiler (trying to open a file that doesn’t exist result in FileNotFoundException)
* Checked exception caught using try-catch block or declared using throws keyword in the method signature

1. Unchecked exception

unchecked exception occur during Runtime typically due to programming error

* **Null pointer Exception**(missing The initialize variable or un-initialized object)
* **ArrayIndexOutOfBoundsException** (accessing an invalid array index)Arithmetic Exception(dividing number by 0)
* **ArithmeticException** (dividing by zero).

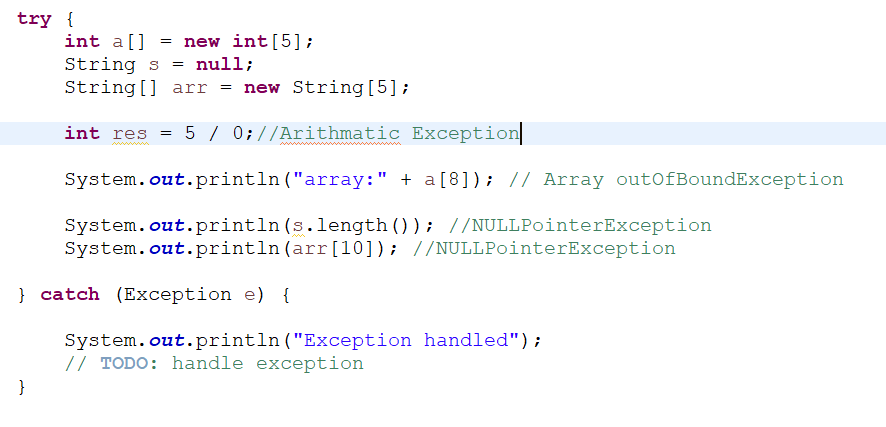
Handling

Try-catch block -for prevent abnormal termination of the program.

A computer code with text

Description automatically generated with medium confidence

A computer screen with text and images

Description automatically generated

A screenshot of a computer code

Description automatically generatedFileNotFoundException->

**Try Catch Finally Block**

* In java we can also use the finally block after try catch block
* Finally block always executed whether there is an exception inside the try block or not (only one time we can used finally block)
* Is it good practice to always use a finally block
* Realtime e.g. Our code execute line by line right so, In a program with 100 lines of code, if an exception occurs midway and we want to ensure a message like 'Code execution ended' is shown, we can use a finally block for that.



start

exception



end

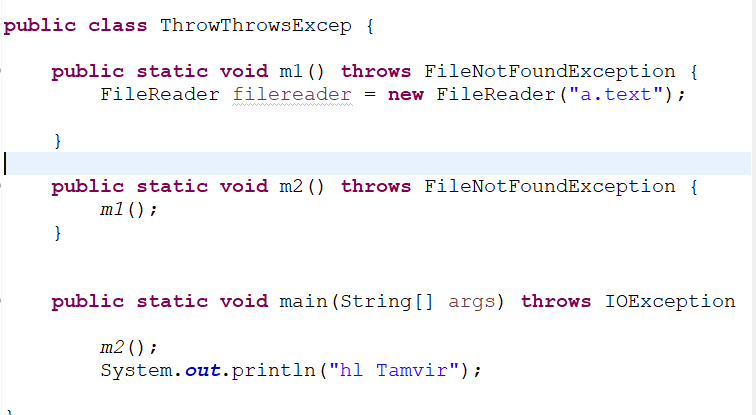
* NOTE- some cases when finally does not executed

1. Use of System. Exit() method //if in program we write Exit()method that time does not executed
2. An exception occurs in finally block
3. The death of thread

**Throw and throws**

**Throw:**

* Throws used inside method body to actually throw an exception
* Purpose is generate exception
* A screen shot of a computer code

  Description automatically generatedTo explicitly throw exception when some error condition arise in program

**Throws :**

* Used in method Signature to declare that method might throw an exception
* To signal to caller that they need to handle or acknowledge potential exceptions that might be thrown by the method

-------------------------------------------NEW-----------------------------------------------------------------

**Generics & Wrapper Class | Auto boxing and Unboxing | Bounded Generics**

Date-16/09/20224

**Wrapper class-**

* A screenshot of a computer

  Description automatically generatedIn java wrapper classes used to convert primitive data types(like int, char) into objects
* Each primitive data type (int ,char ) has corresponding (related) to wrapper class (like Integer ,Character)

**Need –**

1. **Working with collections**

ArrayList,HashSet,HashMap(You cant store an int in an arrayList ,but we can store an Integer )

Eg. ArrayList<Integer > list=new ArrayList<>();

List.add(10); //autoboxing int to Integer

1. **Utility Methods in Wrapper Classes**

* Wrapper class comes with method for converting types like   
  int num=Integer.parseInt(“1234”); //convert string to int

1. **Autoboxing And Unboxing(Automatic Conversion)-**

* The automatic conversion of primitive types to the object of their corresponding wrapper classes is called Autoboxing (int to Integer)
* Its reduce manual conversion and easy to work with collection
* Unboxing is reverse of Autoboxing wrapper object to primitive

Eg.int num=5;

Integer obj=new; // Autoboxing :int to Integer

Int newNum=obj= //Unboxing :Integer to int

1. **Generics**-

* Generics Means Parameterized type (like array List<T>)
* Using generics it is possible to create classes that work with different data types
* A screenshot of a computer program

  Description automatically generatedClass ,interface or method that operates on parameterized type is a generic entity





1. **Generic method-**

* A screen shot of a computer code

  Description automatically generatedIts similar to generics class we can also create method without declaration in class

1. **Bounded Generic Types-**

* Bounded generic used to restrict the type can be passed to class or method
* A black background with colorful text

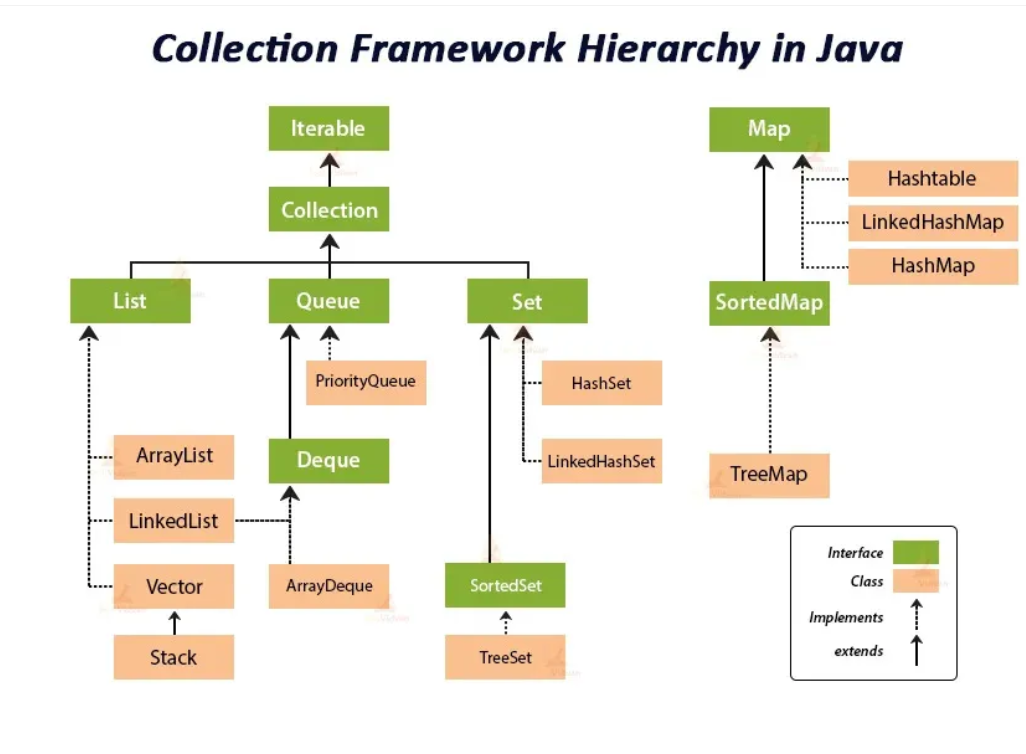
  Description automatically generatedWhen we want to enforce that the generic type must be a specific class or subclass using Extends keyword

**Collection Framework | ArrayList, LinkedList & Collection Interface**

* Java Collection Framework
* Java Collection Interface
* Java List Interface
* Java Array List
* Java LinkedList
* Java stack

1. **Java Collection Framework :**

* **The java collection framework provides a set of interface and classes to implement various DSA.**



1. **Java Collection Interface**

* Collection interface is root interface of java collections framework which represents a group of objects know as elements
* Collection interface is a part of java.util package

Methods in collection interface- add(),remove, contains(check elements),size(),clear(remove all elements), isEmpty(check empty or not),add All(we merge two list),remove all(intersection),clear(list empty) ,toarray(object to array)

**Iterator**-

* iterator in java is an object that allows you to traverse through collection (like Arraylist, Hashset)
* it provides methods to access and remove elements during iteration
*  process of repeatedly accessing each element in a collection or a sequence, one after another. This can be done using loops (e.g., for, while) or with an Iterator.

A computer code on a black background

Description automatically generated

**Why use an Iterator?**

* Safe removal of elements while traversing.
* Can be used with any type of collection.
* **List->**
* ordered collection (like dynamic array )
* Allow duplicates
* List is Interface and it extends to collection
* Array List ,Linked list, vector, Stack

1. **Array List** – Resizable array (automatically adjust their capacity), allow duplicates, random access, Array List can store **null** values. Due to adding removing TC is O(n)

**Array List <String, Integer> AL=new Array List<>()**;

**Internal working in ArrayList ( new size=oldsize\*3/2+1) e**.g 10->increase by 6 ->10\*3->30/2=15+1=16

**Collection vs Collections**

* Collection is an interface that represents group of Objects
* Collections is an utility class that provides method to operate collection type(like searching ,sorting, modifying)

1. **Linked List:- Linked list is class implementation of List and Deque interface its doubly linked list List<> ls=new Linkedlist<>();**

* Linked list store elements in separate containers called nodes where each contain element and like to next node
* No need to resize like Array list
* A screenshot of a computer

  Description automatically generatedFaster when adding and removing elements
* Same method and programmatic working as Array List

1. **Vector :- vector is resizable array implementation of list interface (Vector<E> vc=new Vector<>());**

* the vector class synchronizes each individual operation. Means whenever we want to perform some operation on vector ,the vector class automatically applies a lock to that operation.
* When one thread is accessing a vector and at same time another thread want to access it get an exception called ConcurrentModificationException
* array List is not synchronized

1. **A screen shot of a computer code

   Description automatically generatedStack:- stack is subclass of vector that implements a LIFO data Structure .it provides method push(),pop(),peek(),empty()**

**A grid of text on a white surface

Description automatically generatedDifference between array list ,Linked list, Vector, Stack**

**Java Queue And Sets🡪**

**Java Queue Interface**

The Queue is interface of the java collection framework provides the functionality of the queue Data structure .it extends collection interface(FIFO)

A screenshot of a computer screen

Description automatically generatedA blue squares with black text

Description automatically generated

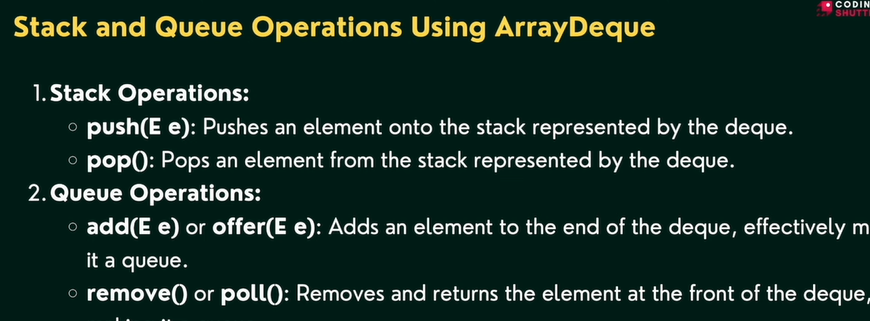
|  |  |
| --- | --- |
| * **offer(E e)🡪**insert elements return true false ( returns false if queue is full) | Boolean Add(E e)🡪insert element but return exception if queue is full |
| * **E poll()🡪** remove elements from queue and return null if q is empty | E remove 🡪remove but return exception if queue is empty |
| * **E peek()🡪** retrieve element which will be remove only showing and return null if queue empty | E element🡪 retrieve element if q empty throw an exception |

1. A screen shot of a computer code

   Description automatically generated**Linked List Queue**🡪

* The linked list class implements to queue Interface and provide method like peek(),poll(),offer()
* It follow FIFO

1. **Array Deque**🡪

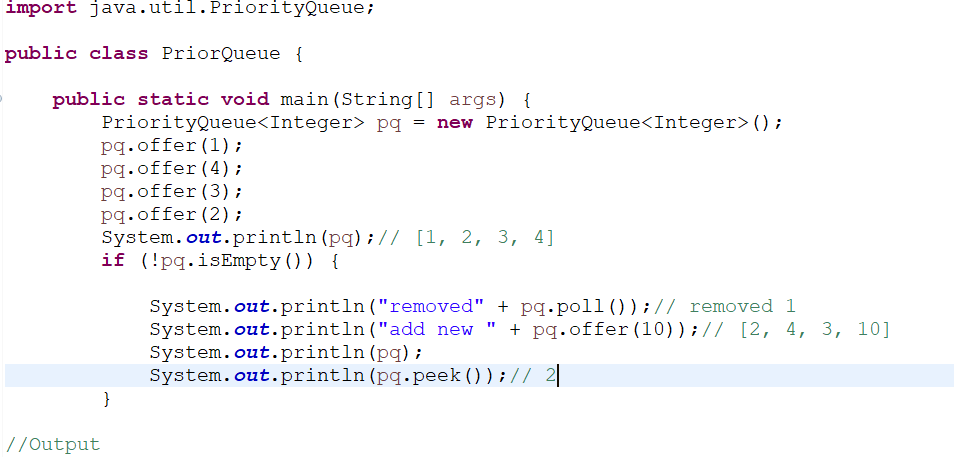
* array Deque is resizable array based implementation of the Deque(Double ended queue)
* it allows elements to be added or removal from both side of queue (pollFirst(),peekFirst(),offerFirst(),getFirst)

A screen shot of a computer code

Description automatically generated

1. **Priority queue**🡪

* According to priority elements are remove and insert
* Elements which is higher priority are removed first if two elements have same priority they are removed in the ordered they were added
* React like minheap means always remove smallest element but we can also custom it like maxheap
* Automatically ordered from smallest element ascending



**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Sets🡪-------------------------------------**

**Java Set interface**

* **The set interface of java collection framework provides the features of the mathematical set in java**
* **It Extend the collection interface it cannot contains duplicate elements**
* **Time complexity is good o(n)**

A diagram of a set

Description automatically generated

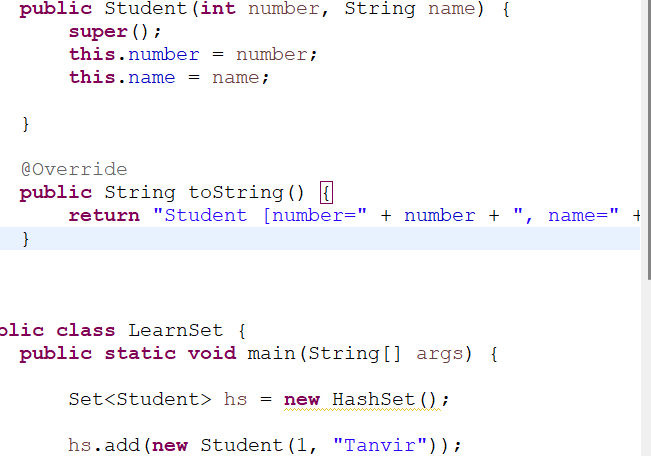
**Java Set Interface**

* Add()🡪add element to set
* addAll()🡪adds all element of the specified collection to the set
* remove()🡪remove specified elements from set
* remove all ()🡪remove all the elements
* retain all()🡪retains all the elements //intersection
* clear()🡪remove all the elements from set
* size()🡪return length of set
* contains🡪check elements pre

**Java Hash set**🡪

* in java hash set is commonly used if we have to access elements randomly
* hash set cannot contain duplicate hench each hash set has a unique hash code
* A screenshot of a computer code

  Description automatically generatedHash code is function like hash code(input)🡪output //hash code has unique identity

**Hash set Using custom objects**

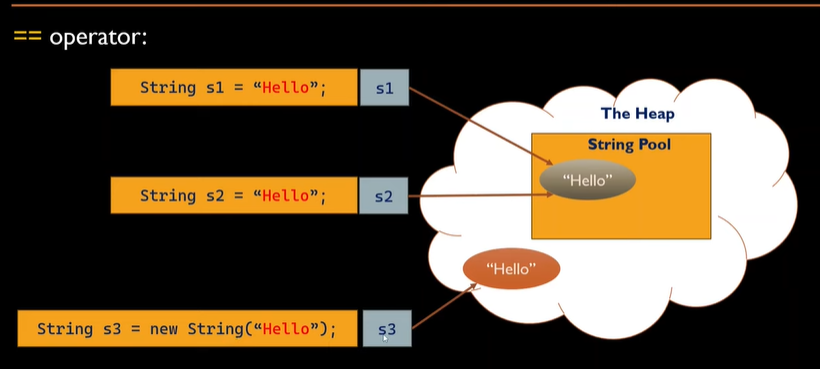
**Hash code and Equals Methods**

**In java both method is used for compare objects like equal or not**

**Equals() and == operator**

**“== Operator”**

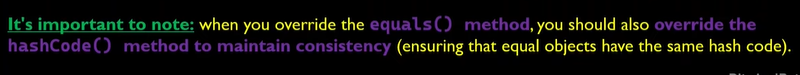
It compares the reference equality of two objects its check two objects references points to the same memory location or same object like below ..

****

* here for non primitive(object) we can see reference (s1,s2) are refer to same String Object(hello) and in same memory location so output is True
* for primitive types (int float, char) it compares actual value of variables

A computer screen shot of a program

Description automatically generated

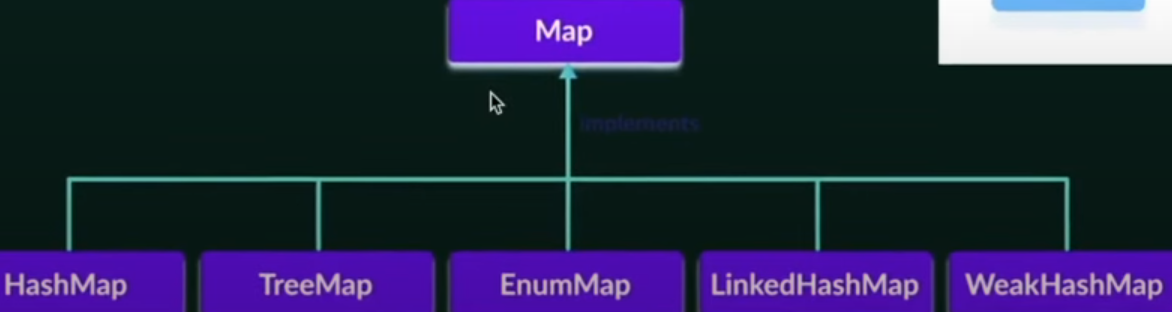


Hashcode() method🡪

* returns int value
* In Java, hashCode() and equals() are two important methods for comparing objects, especially when working with collections like HashMap, HashSet, or Hashtable. They serve distinct but related purposes.
* the hashcode method in java return a hash code value for the object
* it is used by hash-based data structures such as HashMap hash set
* we can create by eclipse

**Map interface🡪**

* In java elements of map are stored in key/value pairs. Keys are unique value associated with individual value
* A diagram of values

  Description automatically generatedA map cannot contains duplicate keys and each key is associated with single value
* Put (k, v)🡪for insert element with key value pair
* PutAll()🡪insert all entries from from the specified map to this map
* PutifAbset(k, v)🡪if absent keypair added
* Get(value or key)🡪get values
* getorDefault(“key” defaultvalue)🡪if value is not present then get defaultvalue
* containsKey(k)🡪checks the specified key is present or not
* containsValue()🡪to check value
* remove()🡪to remove key and using this we can remove key pair
* keyset()🡪we get set of key\
* values()🡪we get set of value
* entrySet()🡪all key and value in present in map
* all complexity O(n)

A screenshot of a computer program

Description automatically generated

**Comparable and comparator :🡪**

* To sort the custom object like student or employee, we need to provide explicit sorting logic
* To achieve this ,java provides the comparable and comparator interfaces.comparable and comparator interfaces in java allow us to define custom sorting behavior for objects,including sorting based on multiple data member