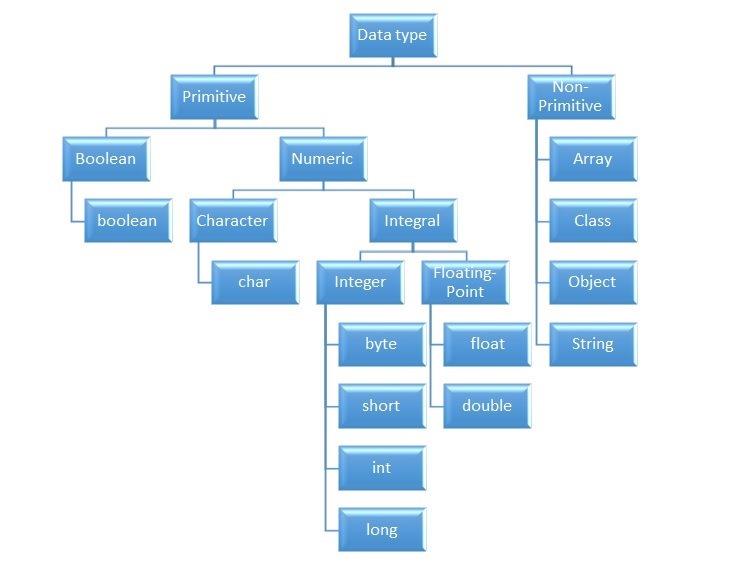
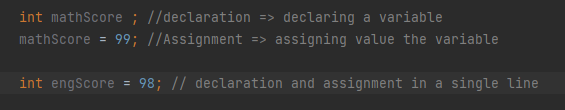
* Java naming conventions:
  + Project name should be all lowercase
  + Package name should be all lowercase
  + Class name should be Pascal case
  + Variable names and method names should be Camel Case
  + What is Pascal case: NumberOfDonuts = 34
  + What is Camel case: numberOfDonuts = 34
* **Primitive Data types**:
  + Byte
  + Short
  + Int
  + Long
  + Double
  + Float
  + Boolean
  + Char
* **Non Primitive Data type**
  + Array
  + Class
  + Object
  + String
* 
* **Byte:**
* A byte in Java is 8 bits. It is a primitive data type, meaning it comes packaged with Java. Bytes can hold values from -128 to 127.
* **Short:**
* The short data type is a 16-bit signed two's complement integer. It has a minimum value of - 32,768 and a maximum value of 32,767.
* **Int:**
* An integer in Java is a memory location that can hold an integer, a positive or negative non-decimal number. It is denoted by the keyword, 'int'..
* **Long:**
* The long keyword is a data type that can store whole numbers from -9223372036854775808 to 9223372036854775808.
* **Float:**
* The float data types are used to store positive and negative numbers with a decimal point,
* **Double:**
* The key difference between a float and a double in Java is that a double can represent much larger numbers than a float. Both data types represent numbers with decimals, but a float is 32 bits in size while a double is 64 bits. A double is twice the size of a float
* **Boolean:**
* A Boolean data type can only have two values: true or false. It is like a yes/no or on/off - a very key component of any programming language.
* **Char:**
* The char keyword is a data type used to store a single character. A char value must be surrounded by single quotes, like 'A' or 'c’

* **Non Primitive Data types:**
* **Arrays:**
* It is a non-primitive data type used to store multiple values of the same data type. Elements of the array can be accessed by the index ranging from 0 to array length - 1. We can use for loop and for-each loop for looping through all the array
* **Class:**
* A class is a data type that restricts access to its data to a set of procedures. These procedures control the ways that an instance of a class (an object) is initialized, accessed, and finally deleted when it is no longer needed**.**
* **Object:**
* The Object data type can point to data of any data type, including any object instance your application recognizes. Use Object when you do not know at compile time what data type the variable might point to. The default value of Object is Nothing
* **String:**
* The String class provides numerous methods for string manipulation. Some commonly used methods include:
* length().
* charAt(int index). Returns the character at the specified index.
* toLowerCase(), toUpperCase().
* trim().
* replace(char oldChar, char newChar
* concat()
* equals()
* **Step 1: To store data**
  + Every program that we write in Java, is some logic around data
  + If we are writing a program on a student management system, the data could be student ID, name, dob, subjects, scores, etc
  + This data can be stored using one of the following
    - *Primitive variables*
    - *Wrapper classes (or Non-Primitive variables)*
    - *Arrays*
    - *Collections*
    - *Strings*
  + **Primitive Variables**
    - What are primitive variables?
    - 8 primitive data types
      * A screenshot of a computer

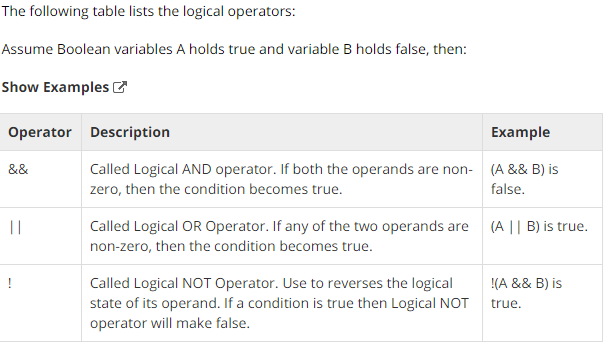
        Description automatically generated
    - Remember they are lowercase
    - How to declare them? How to assign values to them?
      * 
  + Wrapper classes
    - Why do we need wrapper classes when we already have primitive variables?
      * To convert primitive variables into **objects**, that is, to give object form to a data type.
      * To provide several utility methods which are required for primitive data types
    - How to declare them?
      * A number and symbols on a black background

        Description automatically generated
    - How do you assign values to them?
  + Arrays
  + All primitive/nonprimitive data types can store a single value
  + If we want to store multiple values, instead of creating multiple variables, we can use an Array
  + Array can hold multiple values of the same data type. They can store values of the same type as int, long, boolean, etc
  + How to declare an Array? How to initialize/assign values to an Array?
    - A screenshot of a computer program

      Description automatically generated
  + How to access values present in the array?
    - A screenshot of a computer screen

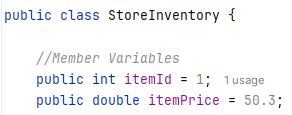
      Description automatically generated
  + Data types of Arrays:
    - We can create an array of primitive variables/data types
    - We can create an array of wrapper objects
    - We can create an array of Strings
* Step 2: To Operate the Data present in Variables (from Step 1)
  + Arithmetic Operators
    - A screenshot of a computer

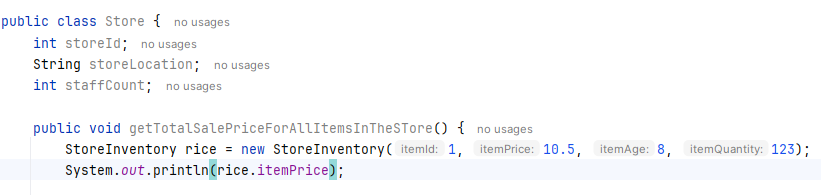
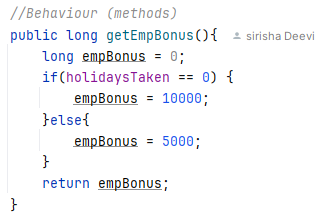
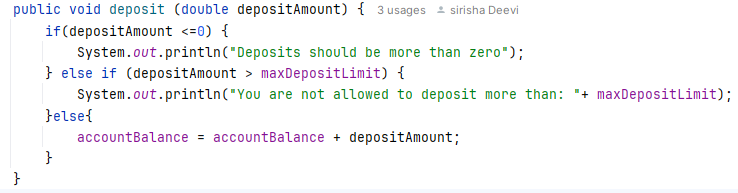
      Description automatically generated
  + Relational Operators
    - A screenshot of a computer

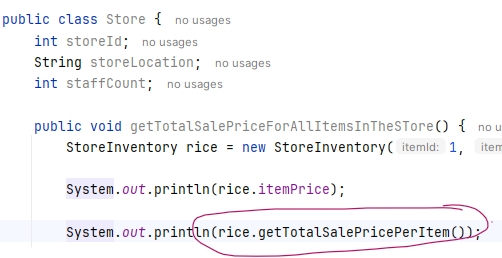
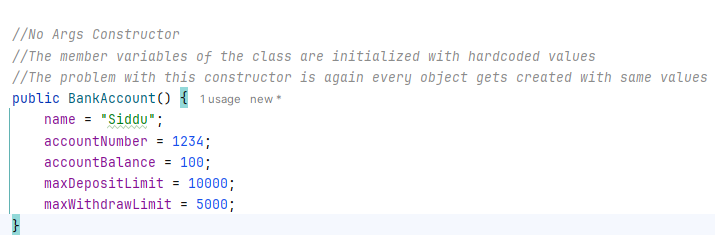
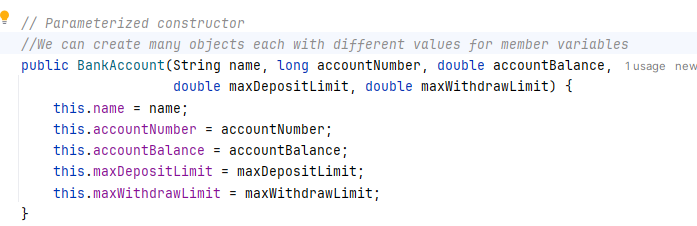
      Description automatically generated
  + Logical Operators
    - 
* Part 5 : Class and Object basics
  + Object Oriented Programming
  + Why Is Java called Object Oriented?
  + Every program or class that we wite in Java is around some real-world object
  + These objects have data and behaviour
  + Examples of Objects
    - Employee
    - Organisation
    - Train
    - Reservation
    - Student
    - Flight
    - Stock
    - Car
  + Each Object Has:
    - Data
    - Behaviour
  + What is a class?
    - Template or blueprint which provides specifications of its objects
    - Class explains the data and behavior of its objects
  + What is an Object?
    - Instance of a class
    - To Instantiate means to create new object of class
    - Class doesn’t have any memory associated with it. Only when the Object is created, memory is allocated to the object
    - Class is like a building plan whereas Object is a real building built using the building plan

* + Class ExampleA screenshot of a computer program

    Description automatically generated
* Members of class:
  + - Data / Member Variables / Instance Variables
    - Behaviour / Member Methods / Instance Methods
    - Constructors
* Members of class: Member Variables
  + Also called as Instance Variables
  + Declaring Member Variables
    - Name
      * Variable has a name
    - Data Type
      * Variable should have data type declared
    - Access Specifiers
      * Public, protected, private, default
      * Discussed in later sections
    - A screen shot of a computer code

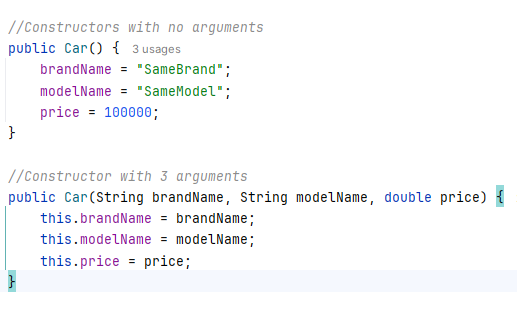
      Description automatically generated
  + Initializing member variables
    - Direct Initialization
      * 
      * The disadvantage with this Initialization is every object we create will have same values
    - Initialization using Constructors:
      * Check Constructors section below
  + Accessing Member variables
    - Within the Same class
      * The member variables of a class can be accessed directly inside member methods of the same class
      * In the flowing example, the member variables “itemAge” and “itemPrice” are directly accessed inside the member method “getFinalItemPrice()”
      * A screenshot of a computer program

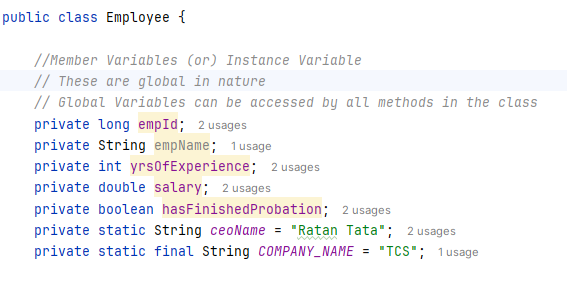
        Description automatically generated
    - Within in different class
      * A member variable of a class can only be accessed using an object inside the member method of another class
      * In the following example, the member variable “itemPrice” of the class “StoreInventory” is accessed inside the member method of another class “Store” using the object “rice”
      * 
      * Variable can be accessed using object
    - Using getter and setter methods
      * See Getters and Setters section below
* Members of class: Member Methods
  + Methods are a block of code which result in some behaviour
  + Example method
    - 
    - getEmpBonus() is a method which calculates employee bonus
  + Method Syntax
    - Name of the method
      * getEmpBonus()
    - Return type of the method
      * The above method returns a “long” type value
      * If method doesn’t return any value, then return type is “void”
    - Parameters/Arguments of Methods
      * A method can have zero or more Parameters or arguments
      * Above method is an example of zero parameters/arguments
      * Example of method with one argument
      * 
    - Method body
      * Method body is a block of code between the brackets { }
    - Method accessors
      * Private, protected, public, default
      * Also called as access modifiers
      * Discussed in later sections
  + Accessing methods of a class
    - Within the Same class
      * A method can be accessed inside another method of same class directly without using object
      * In the following example, the member method “getFinalItemPrice()” is accessed directly inside “getTotalSalePricePerItem()”
      * A screen shot of a computer code

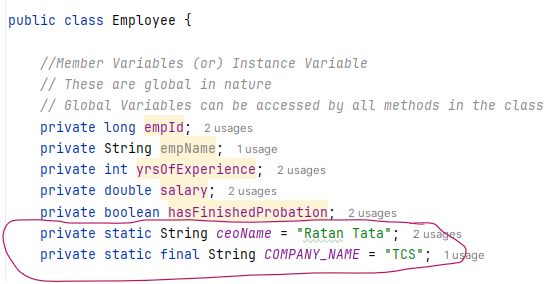
        Description automatically generated
    - Within a different class
      * Inside a member method of a class, a member method of another class can only be accessed using the object
      * In the following example, “getTotalSalePricePerItem()” method which belongs to “StoreInventory” class is accessed using the “rice” object inside the member method of “Store” class
      * 
* Members of class: Constructors
  + What is a constructor?
    - A constructor in Java is a **special method** that is used to initialize objects.
    - The constructor is called when an object of a class is created.
    - It can be used to set initial values for member variables:
    - Initialization=> Giving initial values to variables
  + Name:
    - Name of the Constructor is same as Class name
  + Return type
    - Constructors doesn’t have return type
  + Parameters
    - Constructors may or may not have parameters
    - Constructor without parameters
    - Constructor with parameters
  + Constructor without parameters
    - Following is an example of constructor without parameters
    - We are doing hardcoded initialization of member variables inside the constructor
    - 
    - Creating objects using this constructor:
      * 
    - Disadvantage using this approach is that every object we create would have the same values for the member variables
    - For example, both ***sidAccount*** and ***ananyaAccount*** objects would have same name, accountNumber etc
  + Constructor with Parameters
    - Also called Parameterized constructor
    - The following constructor is taking parameters
    - 
    - 
    - We can create multiple objects where each object has different values for member variables
  + Default Constructor:
    - By default, every class has a default no argument constructor which is invisible
    - And if we write a constructor with no arguments, the compiler does not create a default constructor.
  + Constructor Overloading:
    - The concept of having multiple constructors in the same class is called Constructor

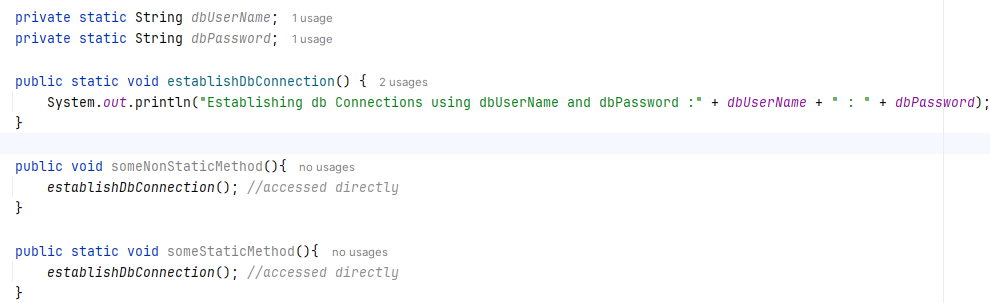
Overloading

While creating a new Object, the constructor with matching parameters is picked automatically

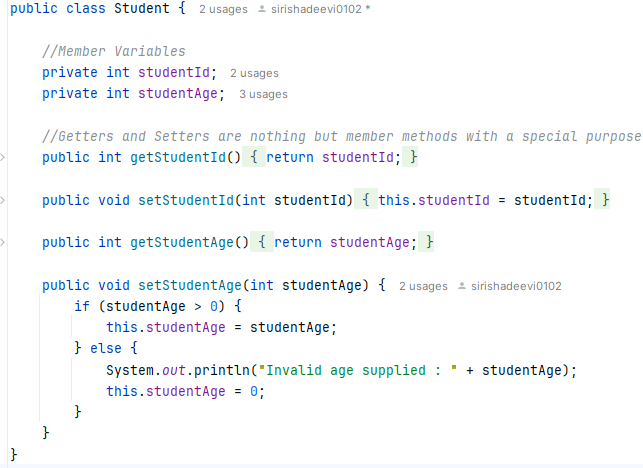


* Variable types (Not Variable Data types)
  + Member Variables / Instance Variables
  + Local Variables
  + Static Variables
* Variable types: Instance Variables
  + These are Member variables of the class
  + These are the variables declared in a class and outside any method present in the class
  + They are globally accessible. they can be accessed inside the member methods of the class
  + We need to provide access specifiers like public, private, protected and default
  + Instance variables need not be initialised explicitly. If not initialized, the following default initial values will be assigned
    - String => null
    - byte, short, int, long => 0
    - boolean => false
    - float,double => 0
    - char => ?
  + 
* Variable types: Local Variables
  + These are the variables declared inside methods
  + They are available locally only within the methods and can’t be accessed outside in any other method
  + Local variables don’t have access specifiers like public, private, protected and default
  + Local variables should be initialised with some values
  + A screenshot of a computer program

    Description automatically generated
* Variable types: Static Variables
  + These are member variables whose value remains same for all the objects of the class
  + If a member variable value is same for all the objects of the class, then those variables can be declared as static variables
  + In the following example, the variables “ceoName” and “COMPANY\_NAME” are the same for all objects of the Employee class. So, they can be declared static
  + 
  + Static variables are also called class variables
  + One copy per class irrespective of number of objects created
  + Created when program starts and before **first** object of class is created
  + Accessing a static variable
    - Inside the member methods of the same class
      * Can be accessed directly
    - Outside the class
      * They can be accessed using class as well as the object. It’s always preferable to access Static variables using class name
      * In the following example, the static variables “ceoName” and “companyName” are accessed using the class “Employee” (but not using the object of “Employee” class)
      * A red circle with black text

        Description automatically generated
  + Static Constant variables:
    - The value of the static variable “ceoName” may change, which means its not a constant. but still its value will be the same for all objects of the class
    - However, the static variable “COMPANY\_NAME” never changes, so it can be declared as final too
    - If a variable is “private static final”, Then it’s called as static constant and its name should be all upper case
* Static Methods:
  + These are member methods which are declared as static
  + Static Methods are also called as Class level methods.
  + How to access a static method?
    - In the same class
      * It can be called directly in the member methods(both static or non-static) of same class
      * 
    - From a different class
      * From a different class, a static method could be accessed using the class name
      * In the following example,”establishDbConnection()” is a static method present in “Emp” class
      * A close-up of a computer screen

        Description automatically generated
  + Static methods can only access a static member variable. It can’t access non static variables
    - 
  + Main method
    - Java program processing starts from the main() method which is a mandatory part of every Java application
    - Main method is the entry point for execution.
    - A screen shot of a computer code

      Description automatically generated
    - **Public**: because it should be accessible from everywhere
    - **Void**: Because it doesn’t return anything.
    - **Static**:
      * There should be only one entry point of execution. Since static is one per class even though multiple objects are created for the class containing the main method, only one main method exists among all objects.
      * If the main method is not declared static, when multiple objects are created for the class containing the main method, multiple main methods can be called one per object which means multiple entry points of execution or multiple main methods running at time which is not acceptable.
    - **(String [] args):**
      * Main method takes a String array as argument
      * Sometimes we might want to pass information into a program when you run it. This is accomplished by passing command-line arguments to main( ).
      * A command-line argument is the information that that is supplied from the command line when it is executed. They are stored as strings in the String array passed to main( ) method
      * In intellij, we can pass these value from run configuration
* Getter and Setter methods
  + These are the member methods to get and set the values of member variables
  + Getter methods are used to read the value of a member variable
  + Setter methods are used to update the value of a member variable
  + Here is an example on how to use the getter and setters
  +  A screenshot of a computer program

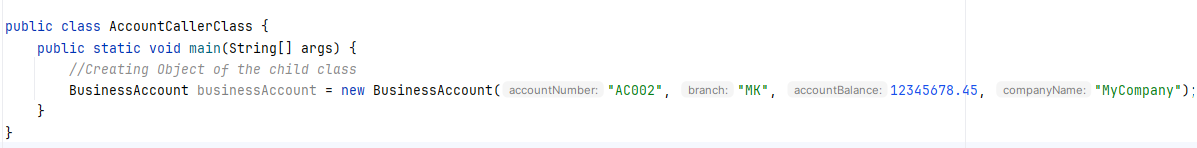
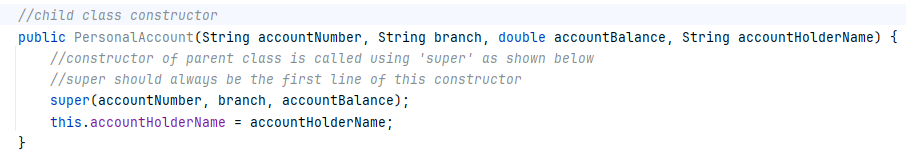
    Description automatically generated
  + Use of getters and setters
    - Use 1: To Restrict access of Data to other classes
      * To have more controls of the data (variables) present in class, we should declare them as private and give access using public getter and setter methods
      * IN the above example, studentId, studentAge are declared private. So they can’t be directly accessed using the object .i.e. ananya.studenId and can only be accessed using the getter and setter methods ananya.getStudentId(), ananya.setStudentId(1)
    - Use 2: To have more control over data modification
      * In the above example, the setStudentAge() method is doing an additional check before updating the studentAge variable. If the setStudnetId() is not used, the studentAge could be assigned with negative values also which is not acceptable
    - Use 3: Encapsulation
      * Discussed in later section
  + Another example of getter and setters
    - A screenshot of a computer

      Description automatically generated
* Part 6: OOPS Concepts
  + Any Object-oriented programming language supports following features
    - Inheritance
    - Encapsulation
    - Polymorphism
    - Abstraction
* OOPS Concepts: Inheritance
  + Inheritance is the process where a class acquires/Inherits members (Variables and methods) of another class.
  + Why to inherit a class?
    - Code reusability is the main advantage of Inheritance
    - When a class A inherits another class B,
      * B is called as parent class
      * A is called as Child class or Sub class
      * Child class A inherits members (data and methods) of class B
      * This allows class A to reuse data and methods of class B
  + Inheritance is achieved by using ***“extends”*** keyword
    - 
  + In the following example,
    - ***PersonalAccount*** class ***extends*** ***Account*** class
    - ***Account*** class is the parent class
    - ***PersonalAccount*** class is the child class or sub class
    - ***PersonalAccount*** inherits all (except private) data and methods from ***Account*** class

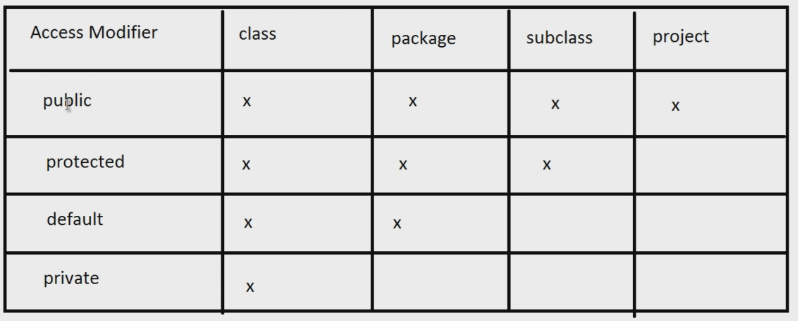
A screenshot of a computer

Description automatically generated

* + - A screenshot of a computer code

      Description automatically generated
    - 
  + How constructor works in case of Inheritance
    - Since Child class Inherits member variables of the parent class, the constructor of the child class should make sure to initialize the member variables of its own as well as the member variables of Parent class
    - This can be achieved using ***“super”*** keyword
    - 
    - In the above example, the child class “PersonalAccount” has only one variable “accountHolderName”. But, since it also inherits accountNumber, branch, accountBalance from parent class, the constructor in the “PersoanlAccount” is passing those three values to parent class constructor using “super” keyword
    - Super should be the first line in the child class constructor
  + A class can be extended multiple classes (A parent can have multiple children)
  + IS-A relation vs HAS-A relation
    - There are two ways to access members (data and variables) of a class in another class
    - IS-A relation (Inheritance)
      * Inheritance is IS-A relation
      * From above example, PersonalAccount IS-A Account
      * Here, child class is inheriting the members of parent class
    - HAS-A relation (Association)
      * This is achieved by creating the object of a class in another class
      * Using the object, we can access members of that class in this class
* OOPS Concepts: Encapsulation
  + Encapsulation is capsuling data and methods into a single unit
  + Encapsulation is also referred to as data hiding. Data hiding is basically not hiding data. It about protecting data from accidental manipulation
  + Encapsulation is achieved using getters and setters and access modifiers
  + To achieve Encapsulation, implement the following two steps
    - Declare the variables private
    - Create public getter and setter methods to read and modify the values of variables
  + In the following example, IN the above example, studentId, studentAge are declared private. So they can’t be directly accessed using the object .i.e. ananya.studenId and can only be accessed using the getter and setter methods ananya.getStudentId(), ananya.setStudentId(1)
  + A computer screen shot of a program

    Description automatically generatedA screenshot of a computer program

    Description automatically generated
  + Access Modifiers
    - There are four access modifiers
      * Private
      * Protected
      * Public
      * Default
    - 
    - Access Modifier: Private
      * Any member (variable or Method), If declared private,
        + It is accessible

only with in the same class

* + - * + It’s not accessible everywhere else

In any other class in the same or different package

In child class

* + - Access Modifier: Protected
      * Any member (variable or Method), If declared protected
        + It is accessible

within the same class

within the child class present in any package

within another class in the same package

* + - * + It is not accessible

Within the class present in different package

* + - Access Modifier: Default
      * If we don’t provide any access modifier (private, protected and public), then it is treated as default
      * Any member (variable or Method), If declared default,
        + It is accessible

Within the same class

Within another class in same package

* + - * + It is not accessible

Within Sub class

Within another class present in different package

* + - Access Modifier: Public
      * Any member (variable or Method), If declared public,
        + It is accessible everywhere

Within the same class

Within different class in any package

Within child class in any package