CLASS LOADING PROCESS OBJECT LOADING PROCESS EXAMPLE PICTURE OF OOPS CONCEPTS

WHAT IS CLASS ?

- CLASS IS A TEMPLATE OR BLUEPRINT OF AN OBJECT.
- CLAS IS A NAMED GROUP OF PROPERTIES AND METHODS.
- CLASS IS ALSO KNOWN AS COLLECTION OF DATA.
- WE CAN CREATE OWN DATATYPE WITH THE HELP OF CLASS NAME.

IN JAVA A CLASS CAN CONTAIN :
STATES , METHODS , CONSTRUCTORS , BLOCKS..

WHAT IS OBJECT ?

- OBJECT IS AN INSTANCE OF JAVA CLASS.
- ANYTHING WHICH IS EXISTING IN THE REAL WORLD IS KNOWN AS OBJECT.
- EVERY OBJECT WILL BE HAVING STATES , BEHAVIOURS AND IDENTITY.

IN JAVA A OBJECT CAN CONTAIN :
STATE - IT IS REPRESENTED BY ATTRIBUTES OF AN OBJECT .
BEHAVIOUR - IT IS REPRESENTED BY METHODS OF AN OBJECT.
IDENTITY - IT GIVES A UNIQUE NAME TO AN OBJECT AND ENABLES ONE OBJECT TO INTERACT WITH OTHER OBJECTS.

WHAT IS CONSTRUCTOR?

- CONSTRUCTOR IS A SPECIAL TYPE OF METHOD WHICH IS HAVING NAME SIMILAR TO CLASSNAME.
- CONSTRUTOR CANNOT HAVA MODIFIER AND RETURN TYPE.
- WHENEVER A CLASS IS NOT HAVING ANY CONSTRUCTOR JAVA COMPILER WILL CREATE DEFAULT NO ARGUMENT CONSTRUTOR.
- WHENEVER A CLASS IS HAVING ANYONE CONSTROCTOR JAVA COMPILER WILL NOT CREATE ANY OTHER CONSTRUCTOR.

<u>**SYNTAX:**</u>

```
[ACCESS MODIFIER] CLASSNAME([FORMAL ARGUMENTS]){
	//STATEMENTS
}
```

EXAMPLE:

ClassName objref = new ClassName();
Constructor obj1 = new Constructor(200,"string");

WHAT IS CONSTRUCTOR OVERLOADING?

• IN A CLASS HAVING MORE THAN ONE CONSTRUCTOR WITH SAME NAME BUT DIFFERENT FORMAL ARGUMENTS IS KNOWN AS CONSTRUCTOR OVERLOADING.

EXAMPLE:

ClassName objref = new ClassName();
Constructor_Call obj3 = new Constructor_Call(420);
Constructor_Call obj1 = new Constructor_Call(20,123.45,"male");
Constructor_Call obj2 = new Constructor_Call(200,56123.4);

//STATIC MEANS EVERY OBJ HAVE SAME VALUE
EXAMPE :

static int age;
Constructor_Call obj1 = new Constructor_Call(20,123.45,"male");
Constructor_Call obj2 = new Constructor_Call(200,56123.4);
Constructor_Call obj3 = new Constructor_Call(420);
System.out.println(obj1.age);//420
System.out.println(obj2.age);//420
System.out.println(obj3.age);//420

WHAT IS CONSTRUCTOR CHAINING?

• IN A CLASS ONE CONSTRUCTOR CALLING ANOTHER CONSTRUCTOR IS KNOWN AS CONSTRUCTOR CHAINING.
• WE CAN ACHIEVE BY USING **this() and super()** STATEMENT**.**

THIS

• **this** CONTAINS CURRENT EXECUTING OBJECT ADDRESS.
• WE USE **this** KEYWORD TO DIFFERENTIATE WHEN THE LOCAL VARIABLE AND GLOBAL VARIABLE BOTH ARE SAME.
• WE USE **this** KEYWORD ONLY INSIDE THE NON STATIC CONTEXT.

SUPER

• WE USE **super** KEYWORD TO CALL THE PARENT CLASS MEMBERS.
• WE USE **super** KEYWORD ONLY INSIDE THE NON STATIC CONTEXT.

THIS()

• WE USE **this()** TO CALL THE CONSTRUCTOR OF SAME CLASS.

• ALWAYS **this()** SHOULD BE FIRST INSTRUCTION INSIDE THE CONSTRUCTOR BLOCK, WE CANNOT USE **this()** AND **super()** STATEMENT TOGETHER.

SUPER()

• WE USE **super()** TO CALL THE CONSTRUCTOR OF PARENT CLASS.

• ALWAYS **super()** SHOULD BE FIRST INSTRUCTION INSIDE THE CONSTRUCTOR BLOCK, WE CANNOT USE **this()** AND **super()** STATEMENT TOGETHER.

PRINCIPLE OF OOPS CONCEPT/ 4 PILLARS OF OOPS CONCEPT

ENCAPSULATION
INHERITANCE
POLYMORPHISM
ABSTRACTION

WHAT IS ENCAPSULATION?

• THE PROCESS OF BINDING/WRAPPING THE STATES AND BEHAVIOURS OF THE CLASS IS KNOWN AS ENCAPSULATION.

REAL TIME EXAMPLES OF ENCAPSULATION?

• CAPSULE CONTAINS MEDICINES.
• ATM MACHINE CONTAINS MONEY.
• JAVA CLASS CONTAINS VARIABLES AND METHODS.
• LAPTOP CONTAINS APPLICATION.
• APPLICATION CONTAINS MODULES.

HOW TO ACHIEVE ENCAPSULATION IN JAVA?

• WE CAN ACHIEVE ENCAPSULATION WITH THE HELP OF CLASS. (BECAUSE CLASS WRAP THE DATAS).

WHAT IS THE ADVANTAGE OF ENCAPSULATION?

• THE ADVANTAGE OF ENCAPSULATION IS **DATA HIDING**.

WHAT IS DATA HIDING?

• IT IS THE PROCESS OF RESTRICTING THE DIRECT ACCESS TO THE DATA MEMBERS BUT PROVIDING INDIRECT ACCESS WITH THE HELP OF GETTER AND SETTER METHOD IS CALLED DATA HIDING.

PROPERTIES OF GETTER METHOD :

• WE SHOLUD CREATE GETTER METHOD WHEN THE DATA MEMBERS ARE PRIVATE.
• **GETTER METHOD IS USED TO GET THE DATA**.
• GETTER METHOD SHOULD BE NON STATIC METHOD.
• GETTER METHOD SHOULD BE NO ARGUMENT METHOD.
• GETTER METHOD SHOULD HAVE RETURN TYPE IT DEPENDS ON DATA MEMBERS.
• **IF WE WANT TO MAKE THE DATA MEMBERS READABLE, CREATE GETTER METHOD.**

PROPERTIES OF SETTER METHOD :

• WE SHOLUD CREATE SETTER METHOD WHEN THE DATA MEMBERS ARE PRIVATE.
• **SETTER METHOD IS USED TO SET THE DATA.**
• SETTER METHOD SHOULD BE NON STATIC METHOD.
• SETTER METHOD SHOULD BE PARAMETERIZED CONSTRUCTOR.
• SETTER METHOD SHOULD BE VOID RETURN TYPE.

WHAT IS BEAN CLASS OR POJO CLASS?

• CREATE PUBLIC NON-ABSTRACT CLASS.
• IN A CLASS ALL THE DATA MEMBERS SHOULD BE PRIVATE.
• CLASS CONTAINS PUBLIC NO ARGUMENT CONSTRUCTOR.
• CLASS CONTAINS PUBLIC GETTER AND SETTER METHOD.
• HENCE WE SAY IT IS **BEAN CLASS OR POJO CLASS**.

WHAT IS FULLY ENCAPSULATED CLASS?

• IN A CLASS **ALL THE DATA MEMBER SHOULD BE PRIVATE** MEANS WE CAN SAY FULLY ENCAPSULATED CLASS.

WHAT IS PARTIALLY ENCAPSULATED CLASS?

• IN A CLASS **ANY ONE DATA MEMBER IS NOT BE PRIVATE** MEANS WE CAN SAY PARTIALLY ENCAPSULATED CLASS.

WHAT IS RELATIONSHIP IN JAVA?

• THE CONNECTION BETWEEN THE OBJECTS IS CALLED RELATIONSHIP.

2 TYPES OF RELATIONSHIP THEY ARE :
HAS-A-RELATIONSHIP.
IS-A-RELATIONSHIP.

WHAT IS HAS-A-RELATIONSHIP?
- WHENEVER THERE IS A DEPENDENCY BETWEEN THE OBJECTS IS CALLED HAS-A-RELATIONSHIP
- IN THAT WE HAVE **COMPOSITION** AND **AGGREGATION**.

WHAT IS COMPOSITION?
- ONE OBJECT **CANNOT EXISTS** WITHOUT ANOTHER OBJECT IS CALLED **COMPOSITION**.
- WE CAN ACHIEVE COMPOSITION BY USING **EARLY INSTANTIATION**(CREATING ONE OBJECT INSIDE ANOTHER OBJECT IS CALLED EARLY INSTANTIATION).

EXAMPLE:
MOBILE-BATTERY, CAR-ENGINE, HUMAN-HEART, OWNER-EMPLOYEE ETC...

WHAT IS AGGREGATION?
- ONE OBJECT **CAN EXISTS** WITHOUT ANOTHER OBJECT IS CALLED **AGGREGATION**.
- WE CAN ACHIEVE AGGREGATION BY USING **LAZY INSTANTIATION**(CREATING ONE OBJECT REFERENCE INSIDE ANOTHER CLASS IS CALLED EARLY INSTANTIATION).

EXAMPLE:
MOBILE-SIM, CAR-PERSON, HUMAN-BIKE, APPLICATION-LAPTOP ETC…

WHAT IS IS-A-RELATIONSHIP?
- THE CONNECTION BETWEEN THE OBJECTS WHICH IS SIMILAR TO PARENT AND CHILD RELATION IS CALLED **IS-A-RELATIONSHIP**.

WHAT IS INHERITENCE?
- THE PROCESS OF INHERIT THE DATA MEMBERS FROM ONE CLASS TO ANOTHER CLASS IS KNOW AS **INHERITANCE**.

• WE CAN ACHIEVE INHERITANCE BY USING **EXTENDS** AND **IMPLEMENTS** KEYWORD.

WHAT IS THE ADVANTAGE OF INHERITANCE?

• THE ADVANTAGE OF INHERITANCE IS WE CAN ACHIEVE **CODE REUSABILITY**.

PARENT CLASS - BASE CLASS - SUPER CLASS
CHILD CLASS - DERIVED CLASS - SUB CLASS

PROPERTIES OF INHERITANCE :

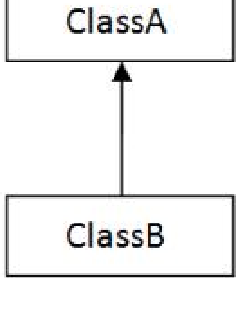
• STATIC VARIABLES AND METHODS WILL BE INHERITED.
• INSTANCE VARIABLES AND METHODS WILL BE INHERITED.
• CONSTRUCTOR AND PRIVATE DATA MEMBERS WILL NOT BE INHERITED.

TYPES OF INHERITANCE :

1. **SINGLE LEVEL INHERITANCE**
2. **MULTI LEVEL INHERITANCE**
3. **HIERARCHICAL INHERITANCE**
4. **MULTIPLE INHERITANCE**
5. **HYBRID INHERITANCE**

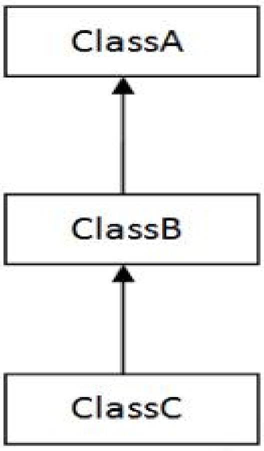
SINGLE LEVEL INHERITANCE :

• IF INHERITANCE IS OF ONLY ONE LEVEL IS KNOWN AS SINGLE LEVEL INHERITANCE.



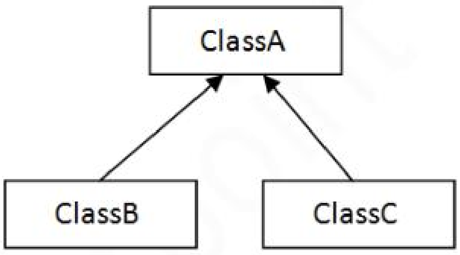
MULTI LEVEL INHERITANCE :

• IF INHERITANCE IS OF MORE THAN ONE LEVEL IS KNOWN AS MULTI LEVEL INHERITANCE.



HIERARCHICAL INHERITANCE :

• IF A PARENT CLASS IS HAVING MORE THAN ONE CHILD IN A SAME LEVEL IS KNOWN AS HIERARCHICAL INHERITANCE.

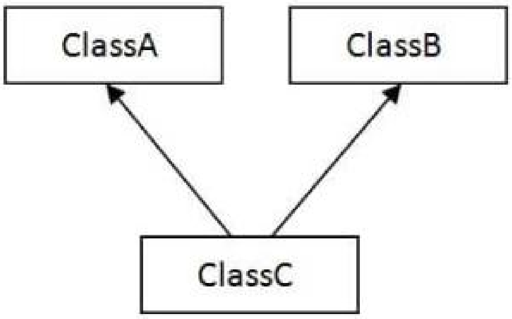


3) Hierarchical

MULTIPLE INHERITANCE :

• IF A CHILD CLASS IS HAVING MORE THAN ONE PARENT IS KNOWN AS MULTIPLE INHERITANCE.

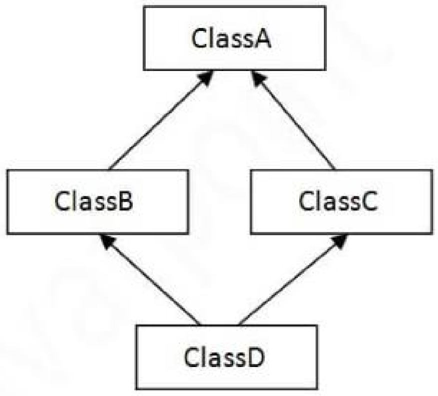
• MULTIPLE INHERITANCE IS NOT POSSIBLE BY USING CLASS, BUT MULTIPLE INHERITANCE IS POSSIBLE BY USING INTERFACE.



HYBRID INHERITANCE :

• IT IS THE COMBINATION OF HIERARCHICAL INHERITANCE AND MULTIPLE INHERITANCE.

• HYBRID INHERITANCE IS NOT POSSIBLE BY USING CLASS, BUT HYBRID INHERITANCE IS POSSIBLE BY USING INTERFACE.



USING CLASS WHY MULTIPLE INHERITANCE IS NOT POSSIBLE?

• THE MAIN REASON IS WHEN CHILD CLASS CONSTRUCTOR WILL CALL PARENT CLASS CONSTRUCTOR CONFUSION ARISES(I.E **CONSTRUCTOR AMBIGUITY**).

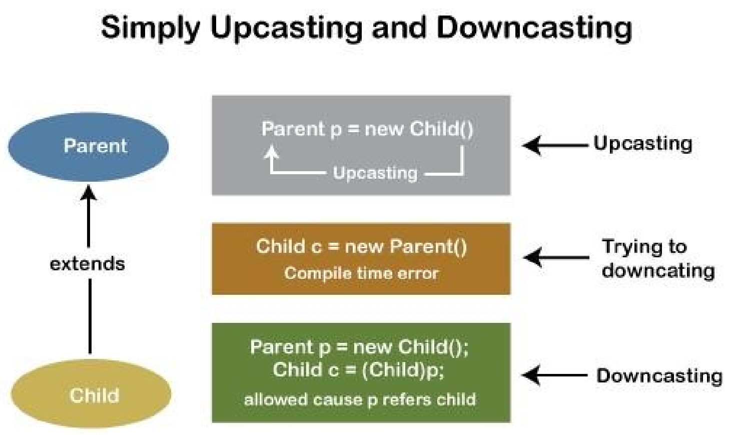
• THIS PROBLEM IS CALLED **DIAMOND PROBLEM**.

• USING **EXTENDS KEYWORD** WE CANNNOT EXTENDS MULTIPLE CLASS.

HOW TO ACHIEVE MULTIPLE INHERITANCE IN INTERFACE?

• IN INTERFACE **CONSTRUCTOR IS NOT PRESENT** SO WE CAN ACHIEVE MULTIPLE INHERITANCE.

<u>**NON-PRIMITIVE TYPE CASTING :**</u>



1. **UPCASTING :**

• THE PROCESS OF STORING CHILD OBJECT REFERENCE IN PARENT TYPE OF CONTAINER IS CALLED AS UPCASTING.

2. **DOWNCASTING :**

• THE PROCESS OF STORING PARENT OBJECT REFERENCE IN CHILD TYPE OF CONTAINER IS CALLED AS DOWNCASTING.

• DOWN CASTING IS NOT POSSIBLE IMPLICIT INSTEAD OF THAT WE CAN DO IT EXPLICIT WITH THE HELP OF TYPE CASTING.

WHAT IS THE ADVANTAGE OF UPCASTING?

• IT IS A COMMON TYPE OF CONTAINER WE CAN STORE ALL THE CHILD REFERENCE.

WHAT IS THE DISADVANTAGE OF UPCASTING?

• THE DISADVANTAGE OF UPCASTING IS WE CANNOT ACCESS THE CHILD OBJECT MEMBERS WITH THE HELP OF PARENT OBJECT REFERENCE.

• JAVA COMPILER ALWAYS TAKE THE DECISION BASED ON THE TYPE.

WHY JAVA COMPILER ALWAYS TAKE THE DECISION BASED ON THE TYPE?

• BECASUSE OF SHADOWING CONCEPT.

WHAT IS THE ADVANTAGE OF DOWNCASTING?

• TO OVERCOME THE DISADVANTAGE OF UPCASTING WE WILL GO FOR DOWNCASTING

WHAT IS CLASSCASTEXCEPTION?

• WITH THE HELP OF PARENT OBJECT REFERENCE CALL THE CHILD DATA MEMBERS WE WILL GET **CLASSCASTEXCEPTION**.

WHAT IS INSTANCE OF OPERATOR?

• WE USE **instance of** OPERATOR TO CHECK INSTANCE MEMBER IS PRESENT OR NOT IN THE GIVEN TYPE.
RETURN TYPE IS BOOLEAN.

EXAMPLE:
ref instance of type

WHAT IS POLYMORPHISM?

• POLY INDICATES MANY MORPHISM INDICATES FORM.

• THE ABILITY OF AN OBJECT IS HAVING SAME NAME BUT DIFFERENT FORMS IS CALLED POLYMORPHISM.

• FOR THE POLYMORPHISM PROGRAM DECISION CAN BE TAKEN BY THE COMPILER AT THE COMPILE TIME AS WELL AS JVM AT THE RUN TIME.

IN JAVA WE HAVE 2 TYPES OF POLYMORPHISM :

1. COMPILE TIME POLYMORPHISM(OR)STATIC BINDING.
2. RUN TIME POLYMORPHISM(OR)DYNAMIC BINDING.

WHAT IS COMPILE TIME POLYMORPHISM?

• IF THE BINDING IS DONE BY THE COMPILER AT THE COMPILE TIME IS KNOWN AS COMPILE TIME POLYMORPHISM.

IN COMPILE TIME POLYMORPHISM WE HAVE 4 TYPES:

1. METHOD OVERLOADING.
2. CONSTRUCTOR OVERLOADING.
3. VARIABLE SHADOWING.
4. METHOD SHADOWING.

METHOD OVERLOADING :

• IN A CLASS HAVING MORE THAN ONE METHOD WITH SAME NAME BUT DIFFERENT FORMAL ARGUMENTS IS KNOWN AS METHOD OVERLOADING.

• IN METHOD OVERLOADING WHICH METHOD HAS TO BE EXECUTED IS DECIDED BY THE COMPILER AT THE COMPILE TIME BASED ON FORMAL ARGUMENTS.

• METHOD OVERLOADING IS APPLICABLE FOR STATIC METHOD AND INSTANCE METHOD.

CONSTRUCTOR OVERLOADING :

• IN A CLASS HAVING MORE THAN ONE CONSTRUCTOR WITH SAME NAME BUT DIFFERENT FORMAL ARGUMENTS IS KNOWN AS CONSTRUCTOR OVERLOADING.

• IN CONSTRUCTOR OVERLOADING WHICH CONSTRUCTOR HAS TO BE EXECUTED IS DECIDED BY THE COMPILER AT THE COMPILE TIME BASED ON FORMAL ARGUMENTS.

VARIABLE SHADOWING :

- IF A PARENT CLASS AND CHILD CLASS HAVING SAME VARIABLE NAME IS CALLED VARIABLE SHADOWING.
- VARIABLE SHADOWING IS APPLICABLE FOR STATIC VARIABLE AND INSTANCE VARIABLE.

METHOD SHADOWING :

- IF A PARENT CLASS AND CHILD CLASS HAVING SAME METHOD SIGNATURE IS CALLED METHOD SHADOWING.
- METHOD SHADOWING IS ONLY APPLICABLE FOR STATIC METHOD .

WHAT IS RUN TIME POLYMORPHISM?

- IF THE BINDING IS DONE BY THE JVM AT THE RUN TIME IS KNOWN AS RUN TIME POLYMORPHISM.

IN RUN TIME POLYMORPHISM WE HAVE 1 TYPE:

1. METHOD OVERRIDING.

METHOD OVERRIDING :

- THE PROCESS OF REPLACING PARENT METHOD BODY BY THE CHILD METHOD BODY IS CALLED METHOD OVERRIDING.

HOW OVERRIDING WORKS?

- **WHICH METHOD IS PARTICULARLY CALLED IS DEPENDENTS ON OBJECT.**
- **STATIC METHOD CANNOT BE OVERRIDE BECAUSE IT IS DEPENDENT ON CLASS.** (COMPILE TIME)
- **INSTANCE METHOD CAN BE OVERRIDE BECAUSE IT IS DEPENDENT ON OBJECT.** (RUN TIME)

WHAT IS ABSTRACTION?

- THE PROCESS OF HIDING THE UNNECESSARY DETAILS(IMPLEMENTATION) AND PROVIDING THE IMPORTANT FEATURES TO THE USER IS KNOWN AS ABSTRACTION.

WHAT IS ABSTRACT CLASS?

- A CLASS PREFIXED WITH ABSTRACT MODIFIER IS CALLED ABSTRACT CLASS.

WHAT IS ABSTRACT METHOD?

• ANY INSTANCE METHOD PREFIXED WITH ABSTRACT MODIFIER AND IMPLEMENTATION IS NOT PRESENT THEN IT IS CALLED ABSTRACT METHOD.

<u>RULES OF ABSTRACTION :</u>

• WHENEVER A CLASS IS HAVING A ABSTRACT METHOD WE SHOULD MAKE THE CLASS AS ABSTRACT CLASS.

• WHENEVER A CLASS IS HAVING INCOMPLETE METHOD **EITHER DECLARED OR INHERITED** WE SHOULD MAKE THE CLASS AS ABSTRACT **OR** WE SHOULD GIVE THE IMPLEMENTATION.

<u>IMPORTANT POINTS OF ABSTRACTION :</u>

• INSIDE THE ABSTRACT CLASS WE SHOULD DECLARE A **STATIC, INSTANCE AND FINAL VARIABLES.**

• WE CAN HAVE CONSTRUCTOR INSIDE THE ABSTRACT CLASS.

• IN ABSTRACT CLASS WE CAN HAVE STATIC METHOD WITH BODY AND INSTANCE METHOD WITH OR WITHOUT BODY.

• WE CANNOT CREATE OBJECT FOR ABSTRACT CLASS.

• IN ABSTRACT CLASS WE CANNOT CREATE A ABSTRACT STATIC METHOD(**BECAUSE STATIC METHOD CANNOT BE OVERRIDE**).

• WE CANNOT MAKE ABSTRACT CLASS AS FINAL.

• WE CANNOT MAKE ABSTRACT METHOD AS FINAL.

WHY WE CANNOT CREATE OBJECT FOR ABSTRACT CLASS?

• IF WE CREATE A OBJECT FOR ABSTRACT CLASS IT WILL SHOW ERRROR,(**BECAUSE IMPLEMENTATION IS NOT PRESENT IN PARENT CLASS**).

• WE CANNOT CREATE OBJECT FOR ABSTRACT CLASS BUT WE CAN CREATE A OBJECT FOR **CHILD CLASS**.

IN ABSTRACT CLASS WHY CONSTRUCTOR CAN ACCEPT AND WHY OBJECT CREATION CAN'T ACCEPT?

• IN JAVA EVERY CLASS CAN HAVE A DEFAULT CONSTRUCTOR BECAUSE TO INITIALIZE THE INSTANCE MEMBERS.

• IN CHILD CLASS WE CALL SUPER CLASS CONSTRUCTOR TO INITIALIZE THE PARENT CLASS MEMBERS.

WHAT IS HIDDEN IN ABSTRACTION?

- WE JUST USE THE METHOD,IN THAT METHOD ALL THE INFORMATION IS HIDDEN FROM US. THAT'S WHY WE SAID HIDING THE UNNECESSARY DETAILS AND PROVIDING THE IMPORTANT FEATURES TO THE USER.

DIFFERENCE BETWEEN ENCAPSULATION AND ABSTRACTION?

- ENCAPSULTATION IS THE PROCESS OF CONTAINING THE INFORMATION.
- ENCAPSULATION MEANS INTERNAL WORKING PROCESS.
- ABSTRACTION IS THE PROCESS OF GAINING THE INFORMATION.
- ABSTRACTION MEANS EXTERNAL WORKING PROCESS.

DIFFERENCE BETWEEN DATA HIDING AND ENCAPSULATION?

- DATA HIDING MEANS MAKE THE DATA MEMBERS AS PRIVATE. **EG PRIVATE INT id;**//THIS IS DATA HIDING.
- ENCAPSULATION MEANS WRAPPING THE DATA AND HIDE THE COMPLEXITY OF THE SYSTEM.USING GETTTER AND SETTER METHOD WE CAN ACCESS THAT IS ENCAPSULATION.

WHAT IS INTERFACE ?

- AN INTERFACE IS A REFERENCE TYPE SIMILAR TO CLASS, THAT CAN CONTAIN CONSTANTS,METHOD SIGNATURES , DEFAULT METHOD AND STATIC METHOD.
- METHOD BODY EXISTS ONLY FOR DEFAULT METHOD AND STATIC METHOD.
- BY USING INTERFACE WE CAN ACHIEVE MULTIPLE INHERITANCE.

<u>IMPORTANT POINTS OF INTERFACE :</u>

- WE CAN'T MAKE INTERFACE AS FINAL.
- WE CAN'T MAKE INTERFACE METHOD AS FINAL.
- WE CAN'T HAVE CONSTRUCTOR INSIDE THE INTERFACE.
- WE CANNOT CREATE OBJECT FOR INTERFACE.