**20-03-2023**

**Alpha tech**

*Write a java program to find the smallest number in an array (tbd)*

*Design a method which accepts the integer array and returns the sum of all the numbers in the array*.

*(AcceptsArrayAndReturnsSum)*

**Object oriented programming language(oops)**

Object oriented programming is a throey / concept . which helps the programmer to represent the real word scenarios through programming .

**Important termenalogies under oops**

1. Class
2. Object
3. Encapsulation
4. Inheritance
5. Polymorphism
6. Abstractions
7. Object : is a real world entity which has properties and behaviours . properties are called as attributes / fields . behaviour is also called as actions or tasks . in java the properties of an object is represented as a variable (non static ) . and behaviours are represented as methods . (non static ). Since java supports object oriented programming . it is called as object oriented programming language .
8. Class : class is a blue print of an object which represents properties and behaviours of the object as variables and methods respectively .(or) class is a tempelate of an object which represents properties as variables and behaviours as methods .

Pillars of object oriented programming :

1. Encapsulation: process of binding properties and behaviours of a method together
2. Inheritance : process of acquiring properties and behaviours of parent to child
3. Polymorphism: ability of an object to exibit more than one form
4. Abstraction : the process of hiding the implementation details

**Constructor :** is a special type of method which is used to initialize the object . everytime an object is created using the new keyword , atleast one constructor is called (it calls default constructor if not other constructors are available ).

There are 2 types of constructors

1. Default constructor
2. Parameteriazed constructor

The constructor name must be seen as a class name . a constructor cannot be abstract , static , final and synchronized .

Note : if there is no constructor in a class compiler will automatically create a default constructor .

Default constructor : default constructor is used to provide default values to the object like zero , null depending upon the data type .

Syntax for constructor:

Access\_specifier class\_name(datatype var1 , datatype var2)

{

Statements;

}

The constructor without a parameter is called default parameter .

**21\03\2023**

**Encapsulation :** it is a process of hiding or binding internal space and requiring all the interactions to be performed to some non static methods .

Internal states : value of variable

Interaction : reading and modifying the value of the variable .

Data hiding: the process of hiding the properties from the outside world is known as data hiding . we can achieve data hiding with the help of private access specifier .

Private : private is a keyword in java which acts as access specifier when a member of a class is made private , it can be used only inside the same class . we cannot use it outside the class .

Note: in order to use the private variable of a class outside a class. We should make use of getter and setter variable .

Getter : getter method is used for reading the value of variable .

Setter: setter method is used for modifying the value of variable . if the private variable should be both readable and modifiable outside the class we should design both getter and setter. we should design both getter and setter method . if the private variable should be only readable outside the class then we should design only getter method . if the private variable should be only modifiable outside the class then we can design only setter method .

Syntax of getter method :

Public datatype\_of attribute get name \_ of \_ attribute

{

Return attribute ;

}

Syntax of setter method :

Public s

**This keyword :** is a predefined object reference variable . it holds address current object .

**Advantages if encapsulation :**

1. We can provide controlled access for the data .
2. With the help of getter and setter method we can perform verification or validation before giving access for the data .
3. Encapsulation improves security .

**Polymorphism :** is the process of performing single action in multiple ways . or . polymorphism is the ability of method to behave directy based on the object it is upon . or . a statement executed multiple times generating different results is also called as polymorphism.

there are 2 types of polymorphism

1. Compiled time polymorphism
2. Run time polymorphism
3. Compiled time polymorphism : the link or bind between method call statement and method implementation happens during compilation of the program. Therefore its called compiled time polymorphism .
4. Run time polymorphism :

**23/03/2023**

**Inheritance :** process of child acquiring for properties and behaviours of its parent is called inheritance .

Example: developer is a employee . in the above example employee is the parent and developer is the child

Developer:

String name;

Int eid;

Double salary;

String designation;

Work();

Code();

Class

Testengineer:

String name;

Int eid;

Double salary;

String designation;

Work();

Write test class();

Class

HR:

String name;

Int eid;

Double salary;

String desigination;

Work;

Recruit;

Class

In the above example developer ,test engineer and HR class have some properties and behaviours(same variables and methods) inside them, that means there is a repetition of code we can avoid this by using second principle of object oriented programing ie inheritance .

Example sent by tatte in watsapp

In the example: employee class acts as parents for developer test engineer and HR and it has all the common properties and behavious of developer , test engineer and HR classes . now the child class ie developer , test engineer and HR are going to inherit these common properties and behaviours from employee class therefore we don’t have to rewrite them inside the child classes therefore major advantage of inheritance is code reusability

Standard definition of inheritance : the process of subclass acquiring properties and behaviours from its super class is called inheritance .

We can achieve inheritance between 2 classes with the help of extend keyword.

Example sent by tatte in watsapp

Note:

The subclass will generally have inherited memebers as well as interpreted members.

Inherited members are the members are the members taken from super class

Interpreted members are the actual or unique members of subclass .

Extends keyword : is a keyword in java which is used to achieve inheritance or is – a – relationship between 2 classes

Extends syntax:

Class class\_name extends class name

(subclass) (super class)

Class A

{

Static int a ;

}

Example:

Class B extends A

{

Public static void main(string[]args)

{

Syso(a); o/p = 0 } }

Types of inheritance

1. Single inheritance
2. Multilevel inheritance
3. Hirerchical inheritance
4. Multiple inheritance
5. Hybrid inheritance

**Single inheritance :** inheritance between 1super class and 1 sub class is called as single inheritance or single level inheritance.

Syntax:

Class A

{

}

Class B extends A

{  
}

Example:

Class a

{

Public static void display()

{

}

}

Class B extends A

{

Public static void main ( strings[]args)

{

b obj1= new B()

obj1.display();

A obj2= new A();

} }

**Multilevel inheritance:**  inheritance of more than one level is called multilevel inheritance

Syntax:

Class A

{

}

Class B extends A

{

}

Class C extends B

{

}

Example:

Class A

{

Public static void display()

{

Syso(“display from a”)

}

Class b extends a

{

Public static void work()

{

Syso(“work()function”);

}

}

Class c extends b

{

Public static void main()

{

Obj1= new c();

Obj1.display();

Obj 1 . work();

}

}

**Hierarchical inheritance** : a super class having more than one sub class in the same level is called hierarchical inheritance.

Syntax:

Class A

{

}

Class b extend a

{

}

Class c extend a

{

}

Example:

Class A

{

Public static void display()

{

Syso(“display from a”)

}

Class b extends a

{

Public static void work()

{

Syso(“work() from class b”);

}

}

Class c extends a

{

Public static void main()

{

Obj 1 = new()

Obj1.display()

Obj 1. Work() ---cte

Obj2=newb()

Obj 2=work()

**Multiple inheritance** : a sub class having more than one super class is called as multiple inheritance

Syntax :

Class a

{

Public static void display()

{

Syso(“from a”);

}

}

Class b

{

Public static void display()

{

Syso(“from b “);

}  
}

Class c extends a, b

{

Public static void main()

{

C obj1= new c ();

Obj.display(); ---- cte  
}  
}

Note: multiple inheritance is not allowed in java because of the abiquity of class cannot extend more than one class . if traid it generates compiled time error, therefore using only classes we cannot achieve multiple inheritance in java.

we can achieve multiple inheritance using interface . This problem of multiple inheritance is called as diamond problems

**hybrid inheritance:** the combination of multiple and hierarchy inheritance is called is called as hybrid inheritance

note : since multiple inheritance is not allowed is java hybrid is also not allowed

**method overriding** : is a process of overwriting or changing the implementation of super class method with sub class method implementation during object creation of subclass

rules for overriding a method:

1. Name of the super class method and sub class method should be same .
2. Formal argument of super class method and sub class method should be same .
3. Written type of super class method and subclass method should be same.
4. Access specifier of super class method and subclass method can be same or subclass method access specifier should have higher visibility compared to super class method access specifier .

Note: overriding happens only with non static methods

Class a

{

Public static void display()

{

Syso(“from a “)

}

}

Class b extends a

{  
public void display()

{

Syso(“from b”);

}

Public static void main()

{

B obj1=new b ()

Obj1 . display(); ----error B

A obj2=obj1; ---- upcasting

Obj2.display() ---- from a

}}

**Up casting (implicit type casting):** it is the process of converting subclass type object reference into super class type object reference

**Down casting (explicit type casting):** it is the process of converting super class type object reference into subclass type object reference is called down casting.

note: down casting must be performed with the help of cast operator

**run time polymorphism** : in the run time polymorphism the link or bind between method call statement or method implementation happens during execution of the program . therefore it is called as run time polymorphism. We can achieve run time polymorphism in a relationship or inheritance , up casting , method overriding .

method overriding example

class a

{

Public int add(int a int b )  
{

Return a + b ;

}

Public int add(int a int b int())

{

Return a+b+c;

}  
public static add(double a , double b)

{

Return a + b ;

}

Public static void main ()

{

Add(10,20)

Add(10.5,11.5);

}

}

Compiled time polymorphsim is also called as early binding or static binding

**24/03/2023**

**Package:**

1. package is a collection of related java file .
2. with the help of package we can organise the java files in a systematic manner .
3. with the help of packages searching or accessing a particular file becomes easy.
4. We can create a package with the help of keyword called package

Syntax:

Package package\_name ;

In the above example sign up package under src must be created by the programmer and s1.java file must to saved inside it .

Sign up package under class must be created by compiler and s1 . class file must be saved inside this can be achieved with the help of package keyword.

**Exceptional handling:** the program occurred due to execution of the program is called as exception . generally when exception occurs program execution stops .

We can avoid this and make the program continue execution this process is know as exception handling .

Java provides following exception handling mechanasim to handle the mechanism

1. Try catch block
2. Try with multiple catch blocks
3. Try catch finally block

There are 2 types of exceptions

1. unchecked exception

2. Checked exception

1. unchecked exception :the compiler unaware exception are called for unchecked exception handling or delaying the exception is not mandatory before compiling

2. checked exception : the compiler aware exception are called checked exception . for checked exception either handling or delaying exception is mandatory before compiling otherwise compiler generates compiled time error

Try block : try block should be statement that is responsible for exception whenever the statement is executed . if there is an exception , jvm creates an object of throwble type and try block throws the code

Try with multiple catch block : when there are multiple catch block we can handle the with the help of try with multiple catch block

Try catch final block : final block can be used with try catch block . finally block gets executed in all the following scenarios .

1. Exception not generated
2. Exception generated and handled
3. Exception generated but not handled

The statements which are mandatory must be written inside the final block .

**Abstraction :** is the process of hiding implementation details we can achieve abstraction using

1. Abstraction class
2. Interface

Abstraction methods : a method with only header and no body is called abstract method

Syntax : abstract access specifier

Abstract method must be prefix with abstract keywork

Abstract method should be ended with it .

**Abstract class :** a class prefix with abstract keyword is called as abstract class

**Note:** if a class has only one abstract method then that class must be declared as abstract class otherwise compiler generates compiled time error

Example:

Abstract class a

{

Abstract public void test (); //abstract method

}

**Concrete method :** a method with implementation is called concrete method . an abstract class can have both abstract and concrete method

**When should we declare a class as abstract ?**

If a class has atleast one defined inherited abstract method so then it is mandatory to declare that class is abstract

Example:

Astract class d

{

Abstract public void a1();

}

Abstract class e abstract d

{  
public void a2()

{  
syso(“from a2()”);

}  
}

**Interface :** in a component in java which acts as the blue print of a class . we can create interface with the help of interface keyword

**Syntax :**

Interface interface\_name

{  
}

Interface can have following members

1. Abstract method
2. Static concrete method
3. Static final variable

Example:

Interface I 1

{

}

Note : the abstract method inside interface are by default considered as public and abstract therefore we don’t have to use public and abstract keyword . while creating abstract method inside interface

Interface can have following members

1. Public abstract non static method
2. Public static config method
3. Public static final variable

Interface cannot have constructor and it cannot create object of interface

Inheritance between interfaces : it can achieve inheritance between interfaces with the help of extend keyword

Interface supports all type of inheritance

***Assignments : write the difference between abstract class and interface***

**String :** string is a predefined class in java which belong to java . language package . it is a non primitive datatype . string class extends object class and also implements comparable , character sequence and serializable variable . string is a final class .

**Can we override method of string class ?**

String being the final class cannot be inherited therefore we cannot override methods of string class . string class objects are used to store string literals (anything that return in double quotes )

Class a

{

String str=”hello”;

Syso(str1);

Int count = str1.length()

For (int I = 0 ; is str1 length(),i++)

{  
class ch = str1.charAt(i);

If (ch==’a’|| ch==’e’|| ch==’I’||ch==’o’||ch==’u’)

{

Int vowel = voel + I;

}

} str 2 = str 2 + ch;

If (str.equal(Str2));

{

Syso(“string is for name”)

}  
}