Algorithm : step by step process to achieve an o/p

Designed through : Pseudocode and Flowchart

Pesudocode(falsecode)
Req: add two numbers

Step 1: declare 3 variables a,b,c
Step 2: give i/p for a and b
Step 3: store the result of Step 2 in variable c
Step 4: display the value of c

Datatypes

Primitivedatatypes: byte,short,char,int,long,float,double,boolean
Non-primitive/referenced datatypes: String,Array,objects

```
class Customer{
}
```

1 byte=8 bits
char=1 byte
int,float=4 bytes
long,double=8 bytes

```
int number=100;
char label='A';
String label="Used";
long mobnum=9898989898L;
float price=10.92F;
double discount=2.0;
boolean value=true;
```

```
int x=100;
int value=100;
```

```
double pi=3.14;
float pi=3.14f;
```

Identifiers: labeling the variable(meaningful)

Keywords: reserved word

```
int try=1000;
```

```
class HelloWorld {
   public static void main(String[] args) {
      boolean x=true;
      boolean y=false;
      boolean a=x^y; //true/1 if x and y are diff, false/0 if x and y are same
      System.out.print(a);
   }
}
```

Naming conventions in Java:

class name: PascalCase
var/method: camelCase
pack name : packagename

Control statements

- selection control
- iterative/loops control

selection control: if,if-else

Control statements

- selection control
- iterative/loops control

selection control: if,if-else

simple if-else
==========
```
if(condition){
}
else{
}
```

multiple if
==========
```
if(condition){
}
if(Condition){
}
else{
}
```

else-if ladder
==========
```
if(condition){
}
else if(condition){
}
.
.
.
else{
}
```

nested if
=========
```
if(condition){
  if(condition) {
  }
  else{
  }
else{
}
```

---

iterative/loops control: while,do-while,for

```
while(condition){
  .
  .
  .  //++ or --
}
```

```
public class Tester {
    public static void main(String[] args) {
        int number=5;
        while(number>0) { //entry controlled loop
            System.out.println(number);
            number--;

        }
        /*do {  //exit controlled loop
            System.out.println(number);
```

```
        number--;
      }while(number>0);*/
      }
}
```

```
for(initialization;condition checking;incre/decre){
}
```

```
public class Tester {
   public static void main(String[] args) {
      /*for(int num=0;num<5;num++) {
         System.out.println(num);
      }*/
      int num[]= {10,20,30};
      for(int i=0;i<num.length;i++) {
         System.out.println(num[i]);
      }
      //for-each loop
      for(int x:num) {
         System.out.println(x);
      }
      }
}
```

[Monday 2:44 PM] Kumar S, Ashok

basic principles of OOP:

Encapsulation - data hiding
Abstraction - relevant ideas/information
Inheritance - parent and child class relationship
Polymorphism - different form of actions

Class - blueprint/template

Object - real world entity and it is an instance of Class

```
class Customer{
    //variables or attributes
    int customerId;
    String customerName;

    //methods or behaviors

    public void purchase() {
        System.out.println(customerName+" purchased "+ "with customerid "+customerId);
    }
    public int calculateTotalBill() {
        return 1;  //just an example
    }
}



public class Tester {
    public static void main(String[] args) {
        //<classname> <ref.var> = <new keyword> <classname>;
        Customer c1 = new Customer();
        c1.customerId=10001;
        c1.customerName="Peter";
        System.out.println(c1.customerId);
        System.out.println(c1.customerName);
        c1.purchase();
    }
}
```

[Monday 2:47 PM] Kumar S, Ashok

Types of variables

[Monday 2:47 PM] Kumar S, Ashok

```
class Customer{
    //variables or attributes
    int customerId;  //instance variable
    String customerName; //instance variable
    int discount;

    //methods or behaviors

    public void purchase() {
        System.out.println(customerName+" purchased "+ "with customerid "+customerId);
    }
```

```
    public int calculateTotalBill(int discount) { //local variable
        int price; //local variable
        return 1;  //just an example
    }
}



public class Tester {
    public static void main(String[] args) {
        //<classname> <ref.var> = <new keyword> <classname>;
        Customer c1 = new Customer(); //reference variable
        c1.customerId=10001;
        c1.customerName="Peter";
        System.out.println(c1.customerId);
        System.out.println(c1.customerName);
        c1.purchase();
    }
}
```

[Monday 3:06 PM] Kumar S, Ashok

Method declaration

[Monday 3:06 PM] Kumar S, Ashok

```
class Tester{
    int num1;
    int num2;
    public int add(int num1,int num2) {  //function definition ==> formal parameters/arguments
        return num1+num2;
    }
    public static void main(String[] args) {
        Tester obj=new Tester();
        System.out.println(obj.add(10,20)); //function call ==> actual parameters/arguments
    }
}
```

```
class Tester{
    int num1;
    int num2;
    public int add(int num1,int num2) {  //function definition ==> formal parameters/arguments
        return num1+num2;
    }
    public boolean verifyNumbers() {
        if(num1>num2) {
```

```
            return true;
        }
        else
            return false;
    }
    public static void main(String[] args) {
        Tester obj=new Tester();
        System.out.println(obj.add(10,20)); //function call ==> actual parameters/arguments
        System.out.println(obj.verifyNumbers());
    }
}
```

Constructor:

- special method having same name as that of class name
- invoked automatically whenever an object is been created
- can have any number of constructor(s)
- parameterless and parameterized constructor

```
class Customer {
    int customerId;
    String customerName;
    int discount;

    public Customer() {
        System.out.println("Inside 1st constructor...");
    }

    public Customer(int customerId) {
        this.customerId=customerId;
        System.out.println("Inside 2nd constructor...");
    }

    public Customer(String customerName,int customerId) {
        this.customerId=customerId;
        this.customerName=customerName;
        System.out.println("Inside 3rd constructor...");
    }

}
```

```
public class Tester {
   public static void main(String[] args) {
      Customer c1 = new Customer(); //default parameterless constructor invoked at the back end
//invoke line 9
      Customer c2 = new Customer(1000); //invoke line 13
      Customer c3 = new Customer("Raju",1000); //invoke line 18
   }
}
```

Access modifiers:

- public -> accessible everywhere(same and diff package)
- private -> accessible only within the class
- default -> accessible within the same packages
- protected -> accessible within same packages and sub-class(child class) of other packages

---

```
public class ArrayDemo {
   public static void main(String[] args) {
      //initialize
      char []label=new char[3];
      //declare
      label[0]='A';
      label[1]='B';
      label[2]='C';
      System.out.println(label);
      //access
      for(int i=0;i<label.length;i++) {
         System.out.println(label[i]);
      }
      for(char x:label) {
         System.out.println(x);
      }
   }
}
```