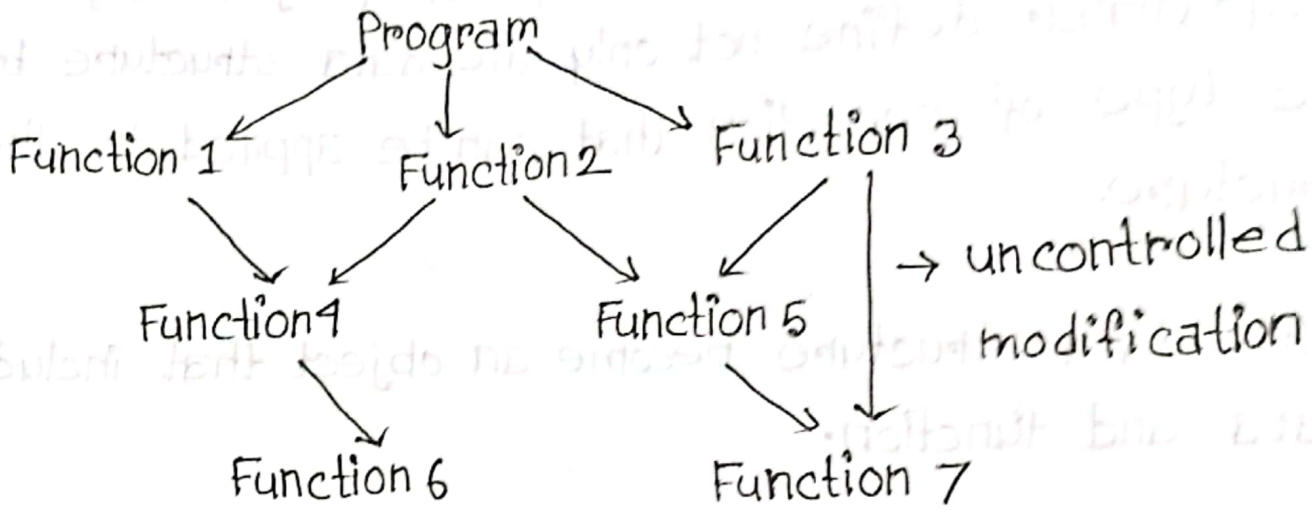
→ POP/OOP

- A program in procedural is a set of instruction.
- The program are divided into subprogram called function.



→ problem of enhancement and maintainability

→ Code reusability

what do we have to do to solve this problem.

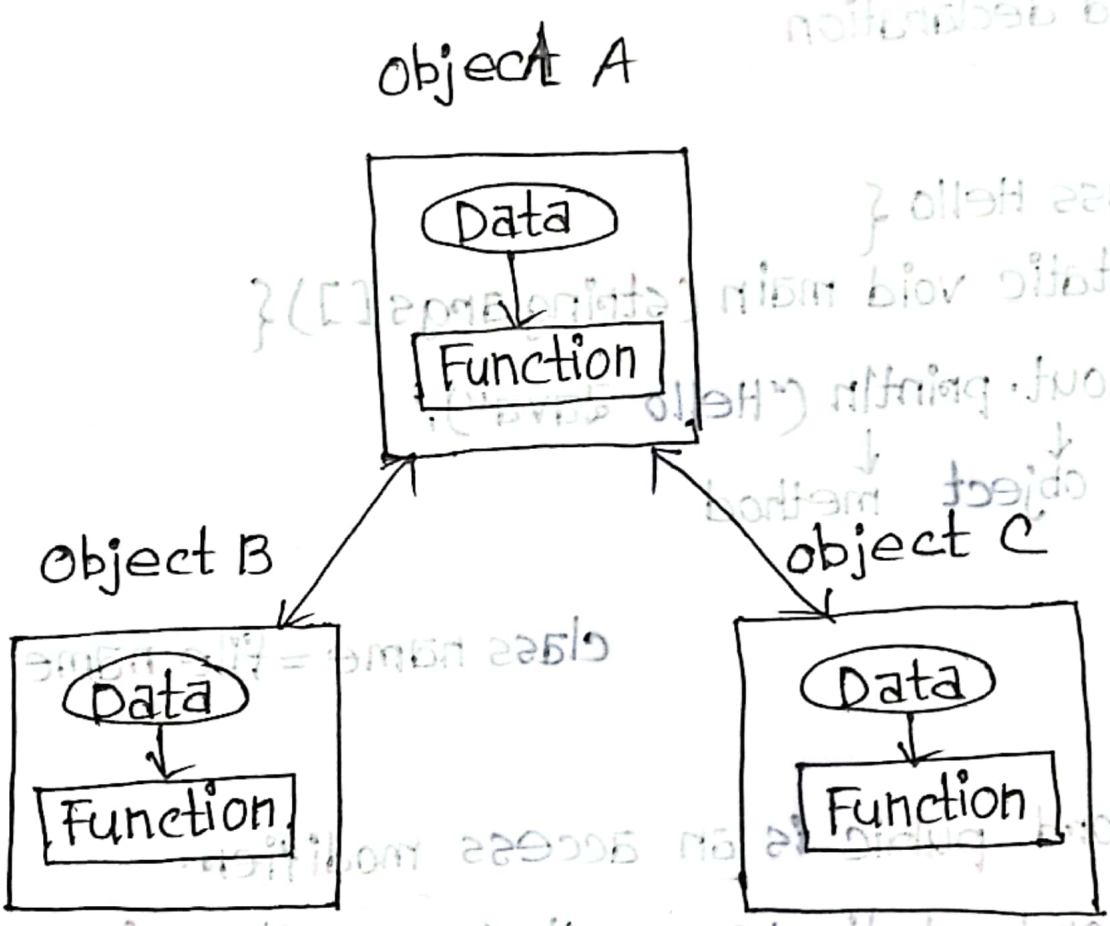
→ Major characteristics:

- more emphasis is on doing things
- It is based on the problem at hand
- Used global variable that lead following problem.
- It will produce ~~p~~ software difficult to maintain
- The risk of unwanted access and modification increases.

CamScanner

- It will not result in reusable software
- It employs top down approach in program design.
- Works well in small system.

→ OOP approach:



CamScanner

• OOP: Object Oriented programming is a programming language in which program are organized data or object rather than function and logic.

→ OOP refers to type of computer programming in which programmer define not only the data structure but also the type of operation that can be applied to the data structure.

→ The data structure become an object that include both data and function.

→ Programmers can create relationship between object.

→ object can inherit the characteristics from other object.

OOP Advantages:

~~Feature of Java:~~

- code reusability
- Data redundancy
- code maintanance
- Security
- Design benefit
- Polymorphism flexibility

CamScanner

→ Basic feature of OOP:

- class
- object
- Inheritance
- Encapsulation
- Polymorphism
- Abstruction

Object:
- Object is an instance of class.
- The object is an entity which has state and behaviour.
- Object has state and behaviour.

State: Represent data (value) of object.

Behaviour: Represent functionality of an object i.e deposit, withdraw.

Example: Pen.

State – colour, brand, size

Behaviour – writing

class:
- A class is a blueprint / prototype from object are created.
- A class is a blueprint, idea and prototype for creating object.

CamScanner

- A class is a group of object which have common properties.

Example: sketch (prototype) of house
- It contains all details about floor, door, windows etc.

- Based on this description, we can create house.
- Here, house is the object.

```
→ class student {
    string name;
    int ID;
    int age;
  }
```

object :-

```
className objectName =
    new className();

student object1 = new student();
object1.name = "Sadika";
object1.ID = "IT20029";
object1.age = "20";
```

Encapsulation:

- The wrapping up of a data and function into a single unit is called encapsulation.
- The data is not accessible outside the class.

MIRCERA
methoxy polyethylene glycol-epoetin beta

CamScanner

Abstruction:

- ~~Reppesent~~ Defines the reppresentation of essential features without including backgrouand details or explanation.

Inheritance: Refers to a process in which object of one class acquires the property of another class.

Polymorphism: It means the ability to take more than multiple form.

Book: Programming with Java (E.Balagurushamy)
Latest Edition

Java - How to program (Deitel & Deitel)
The complete reference Java - H. schilt/

CamScanner

Basic feature of Java/properties

- Compiled and interpreted
- Object Oriented
- Platform independent
- Distributed
- High performance
- Simple and familiar
- Multithreading

Application:

- Mobile application
- Desktop " (swing, Awt, Java Fx)
- Web based " (JEE, servetlet, JSP, ...)
- scientific "
- Gaming "
- cloud ~~baged~~ based "
- Business "

→ Variable
→ Token
→ Data type
→ Identifier
→ Operator

→ Control statement (if, else, switch)
→ loop statement (for, while, do-while)

MIRCERA
methoxy polyethylene glycol-epoetin beta

CamScanner

Token: Token is basically smallest element that can identified by the compiler.

Token
- keyword (int, while, ...)
- Identifier (sum, ...)
- Constant (10, 20, ...)
- String (Sadu, Nadu...)
- Special symbol ({}, (), ...)
- Operator (+, -, ...)

Identifier:
- is a sequence of character.
- are program designed token.
- used for naming, class, method, variable, object, package.....

Keyword: Reserved word / identifier (int, float, double, if, switch.....)

Variable:
- A variable is an identifier that denotes a storage location used to store data value.
- Their value changes during the execution of program
- Every variable must be declared before used.

Rules:
- Variable names may consists of alphabet, digit, underscore and dolar sign.

CamScanner

...amming is ... data or object

- They must not begin with a digit. 9
- Upper case and lowercase are distinct
- It should not be keyword.
- White space not allowed.
- Variable name can be any length.

<u>Data types:</u> Data type specifies the size and type of values that can store in variable.

→ Primitive datatype (integer, character, Boolean, floati
→ Non-primitive " (class, interface, Array, string)

<u>Operator:</u> • An operator is a symbol that tells the computer to ~~pestor~~ perform certain mathematical or logica operation.

- Used in program to manipulate data and variable.

→ Arithmatic Operator → (+, −, /, *, ...)
→ Relational " → (==, !=, >, <, >=, <= ...)
→ logical " → (&&, ||, ! ..)
→ Assignment " → (=, +=, −=, *= ...)
→ Increment/decrement " (++, −−,)
→ Conditional " → (?, :)
· Bitwise " → (&, |)
· Special " → (.)

MIRCERA
methoxy polyethylene glycol-epoetin beta

CS CamScanner

Program structure

- Documentation section
- Package statement
- Import statement
- Interface statement (variable - public static final, method - abstract)
- Class defination
- Main method declaration

→ public class Hello {
    public static void main (string args[]) {
    System. out. println ("Hello Java");
    class object method
  }
}

class name = file name

- The keyword public is an access modifier.
- The keyword static means that a method is accessable and usable without object creation.
- Void means main method does not return any value.

CamScanner

- Polymorphism: Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance

- Compile time polymorphism: Compile time polymorphism is a polymorphism that is resolved during the compilation process. Compile-time polymorphism is achieved by method overloading and operator overloading.

- Run-time polymorphism: Run-time polymorphism is a process in which a call to an overridden method is resolved at runtime rather than compile time.

- How to perform method overloading in Java
  1. Overloading by changing the number of parameters
  2. By changing the datatype of parameters
  3. By changing the datatype sequence

method overloading: In Java, two or more methods may have the same name if they differ in parameters. These methods are called overloaded methods and this feature is called method overloading.

CamScanner

• Method overriding: In Java, method overriding occurs when a subclass (child class) has the same method as the parent class.

1. Method name must same as parent class and child class
2. Return type must be same for all.
3. Final method cannot be overridden
4. Static method cannot be overridden
5. Private method cannot be overridden

• Super keyword: The super keyword refers to superclass (parent) objects. It is used to call superclass methods, and to access the " constructor

1. Super word, used to refer parent class object.
2. used to invoke parent class variable.
3. used to invoke " " constructor
4. " " " " " method

• Final keyword: The final keyword is a non-access modifier used for classes, attributes and methods, which makes them non-changeable (impossible to inherit or override). Final keyword used to restrict the user.

1. Variable
2. Method
3. class

CamScanner

Method overloading

```
→ public class Overload {
    void add (int a, int b) {
        system.out.println (a+b);
    }
    void add (double a, double b){
        system.out.println (a-b);
    }
    void add (int a, int b, int c) {
        system.out.println (a*b*c);
    }
}
public class OverloadTest {
    public static void main (String [] args) {
    Overload ob = new Overload();
    ob.add (7,5);
    ob.add(7.5, 5.5);
    ob.add (1,2,3);
    }
}
```

CamScanner

Method Overriding

```
public class person {
    string name;
    int age;
  void displayInformation() {
    System.out.println("Name:"+name);
    system.out.println("Age:"+age); }
  }

pubic class Teacher extends Person {
    string qualification;

  void displayInformation() {
    System.out.println("Name:"+name);
    system.out.println("Age:"+age);
    system.out.println("Qualification:"+qualification
  }
}

public class Test {
    P.S.V.M(String[] args) {
Person ~~Teacher~~ t1 = new Teacher();
    t1.name = "Sadika";
    t1.age = 21;
    t1.qualification = "BSc in ICT";
    t1.displayInformation(); }
}
```

CamScanner

```
class C{
    int x=10, y=5;
    void math(){
      int z=x+y;
      System.out.println("z="+z);}
    }

class D extends to c{
      int x=20, y=10;
   void math(){
      int z=x*y;
      System.out.println("z="+z);}
    }
public class Overriding{
  public static void main(String []args){
      C ob1 = new C();
      ob1.math();
      D ob2 = new D();
        ob2.math();
      C ob3 = new D();
       ob3.math();}
  }
```

CS CamScanner

Super Keyword

```
1.public class A{
    int x=7;
  }
 public class B extends A{
    int x=13;
  void display(){
      system.out.println("super.x);}
  }
 public class Test{
    public static void main(String []args){
     B ob1 = new B();
      ob1.display();}
  }
```

```
2.public class A{
    void display(){
      System.out.println("Inside A class");}
  }
 public class B extends A{
    super.display();
    void display(){
      super.display();
      system.out.println("Inside B class");}
  }
```

3.
```
public class A {
  A() {
    System.out.println("A's constructor"); }
  }
public class B extend A {
    B() {
      super();
      System.out.println("B's constructor"); }
    }
public class Super {
    P.S.V.M (String [] args) {
  B ob = new (B)(); }
}
```

CamScanner

Abstruction

- Abstruction refers to the representation of essential feature without including background details.
- Abstruction is a process of hiding the implementation details and showing only functionality to user

→ There are two ways to achieve abstruction in java

1. Abstruct class (0 to 100%)
2. Interface (100%)

- Non abstruct method

```
void message(){
  ...
}
```

abstruct method

```
abstruct void message();
```

→

```
abstruct class MobileUser{
    abstruct void sendMessage();
}
public class Rahim extends MobileUser{
    void sendMessage(){
        system.out.println("I am Rahim");}
    }
public class Karim extends MobileUser{
    void sendMessage(){
        System.out.println("I am Karim");}
}
```

CamScanner

```
public class Test {
    public static void main(String [] args) {
        MobileUser ms;
        ms = new Rahim();
        ms.sendMessage();
        ms = new Karim();
        ms.sendMessage(); }
}
```

100% abstract

Or,

```
abstract class MobileUser {
    abstract void sendMessage();
    void call() {
        system.out.println("Call me"); }
}
```

- - - -

- - - -

- - - -

(0 – 100%)

abstract

```
MobileUser ms;
ms = new Rahim();
ms.call();
ms.sendMessage();
```

CS CamScanner

→ abstract class Shape {
    abstract void draw();
  }

class Rectangle extends Shape {
    void draw() {
      System.out.println ("drawing rectangle"); }
  }

class Circle extends Shape {
    void draw() {
      System.out.println ("drawing circle"); }
  }

class Test {
    public static void main(String[] args) { ~~Shape~~ s
      Shape s = new Rectangle();
      s.draw();
      Shape s = new Circle();
      s.draw(); }
  }

CamScanner

```
• abstract class BaseClass {
    abstract void show1();
    void show2() {
      System.out.println ("Concrete method"); }
    }
class childClass extends BaseClass {
    void show1() {                          abstract method
      System.out.println("Overriding"); }
    void show2() {
      System.out.println ("Overriding concrete method");
    }
  }

class Abstraction {
    public static void main(String[] args) {
      BaseClass ob = new ChildClass();
      ob.show1();
      ob.show2(); }
  }
```

CamScanner

Interface

- An interface in the java programming language is an abstract type that is used to describe a behaviour that classes must implement.

```
interface Animal {
    abstract void eat();
}
class dog implements Animal {
public void eat() {
        system.out.println(" Dogs eat");
    }
}
class cat implements Animal {
public void eat() {
        system.out.println("Cats eat"); }
    }

class Test {
  public static void main(string []args]{
   dog d = new dog();
    d.eat();
  cat c = new cat();
    c.eat(); }
}
```

CamScanner

Class	Interface
1. In class you can instantiate variables and create an object	1. In an interface, you can't instantiate variables and create an object.
2. class can contain concrete methods.	2. The interface cannot contain concrete methods
3. The access modifier used with classes are private, protected and public.	3. In interface only one specifier is used public.

→ Package in Java is a machanism to encapsulate a group of class, sub package and interfaces.

→ Package keyword used to create package in java.

→ A class can have only one package declaration.

→ Every class is a part of some package.

Advantage:

- Preventing naming conflict
- better organization
- providing controlled access
- Reusability

CamScanner

• Interface Example:

```
interface Drawable{
    void draw();
  }
Class Rectangle implements Drawable{
    public void draw(){
      System.out.println("drawing rectangle");
    }
  }
class Circle implements Drawable{
    public void draw(){
      System.out.println("drawing circle");
    }
  }
class TestInterface{
    public static void main(String[] args){
      Drawable d;
      d = new Rectangle();
      d.draw();
      d = new Circle();
      d.draw();
    }
  }
```

CamScanner

builtin package:

```
package animal;
import java.util.scanner;
public class dog{
  public static void main(String[]args){
    scanner input=new scanner(system.in);}
}
```

• Types of access modifier:

Modifier	Description
Default	declarations are visible only within the package.
Private	declarations are visible within the class only.
Protected	declarations are visible within the package or all subclasses.
public	declarations are visible everywhere

CamScanner

Interface support multiple inheritance in java

```
interface A{
    void play();
  }
interface B{
    void play();
  }
class C implements A,B{
  public void play(){
      system.out.println("Hello, I am from C");
  }
}
class Test {
  public static void main(String[] args){
      C ob = new C();
      ob.play();
    }
  }
```

We know, multiple inheritance is not supported in the case of class because of ambiguity. However, it is supported in case of an interface because there is no ambiguity. It is because its implementation is provided by the implementation class.

CS CamScanner

• "this" keyword used to refer current class object

(varriable call)

```
class Person{
   string name;
   int age;
 person(string name, int age){
     this.name = name;
     this.age = age; }

 void display(){
     s.o.p ("Name:"+name);
     s.o.p ("Age:"+age); }
   }
class Test{
   public static void main(string[] args){
    person P1 = new person("sadika", 21);
     P1.display(); }

   }
```

→ Same :

(constructor call)

```
  string haircolor;
  ---
  ...

  person(string name, int age, string haircolor){
   this(name, age);
   this.haircolor = haircolor;
   }
```

CamScanner

```
void display(){
    ------
    s.o.p ("Haircolor: " + haircolor);
  }
}
class Test{
  ------
  ------
person P2 = new Person ("Nihal", 24, "black");
 P2.display(); }
}
```

- method call

```
class Person{
  void message(){
      s.o.p("I am message method");
  }
  void display(){
    this.message();
    s.o.p("I am display method");
  }
}
class Test {
    p.s.v.m(String [] args){
   person P1 = new person();
    P1.display(); }
}
```

CamScanner

• Inheritance: The process of obtaining the data members and functions from one class to another class is known as inheritance.

There are five types of inheritance –

- single
- Multilevel
- Multiple
- Hierarchical
- Hybrid

• Single inheritance: In this, a class inherits the properties from a single class.

Example:

```
class Animal{
    void eat(){
     System.out.println("eating");}

  }
class Dog extends Animal{
    void bark(){
        System.out.println("barking");}
  }
class Test{
  public static void main(String args[]){
  Dog d= new Dog();
  d.bark();
  d.eat();}
}
```

CamScanner

Multilevel Inheritance: Where there is a chain of inheritance it is known as multilevel inheritance.

```
class Animal{
 void eat(){
   system.out.println("Eating");}
 }
class Dog extends Animal{
  void bark(){
    System.out.println("barking ");}
  }
class BabyDog extends Dog{
  void weep(){
    system.out.println("weeping");}
  }
class Test{
  public static void main(string args[]){
   BabyDog d = new BabyDog();
   d.weep();
   d.bark();
   d.eat();}
 }
```

CS CamScanner

• Hierarchical Inheritance: When two or more classes inherits a single class, it is known as hierarchical inheritance.

```
class Animal{
 void eat(){
   S.O.P("eating");}
 }
class Dog extends Animal{
 void bark(){
     S.O.P("barking");}
    }
 Class Cat extends Animal{
  void meow(){
        S.O.P("meowing");}
   }
```

```
class Test{
 P.S.V.M(String args[]){
  Cat c = new Cat();
   c.meow();
   c.eat();}
  }
```

• Default Access Modifier: If we do not explicity specify any access modifier for classes, methods, variables etc, then by default the default access modifier is considered.

For example:

```
package defaultPackage;
 class Logger{
     void message(){
        S.O.P("This is a message");}
    }
```

CamScanner

• Private: when variables and methods are declared private, they cannot be accessed outside of the class.

For example:

```
class Data{
    private string name;
class Main{
    public static void main(string args[]){
        Data d = new Data();
        d.name = "Sadika"; }
}
```

• Protected: When methods and data members are declared protected, we can access them within the same packages as well as from subclasses.

For example:

```
class Animal{
    protected void display(){
        S.O.P("Animal"); }
}
class Dog extends Animal{
    P.S.V.M(String args[]){
        Dog d = new Dog();
        d.display(); }
}
```

CamScanner

Public: When methods, variables, classes and so on declared public, then we can access them from anywhere. The public access modifier has no scope restriction.

CS CamScanner