//run time polymorphism

//dynamic method dispatch

class A{

    void read(){

        System.out.println("Read from HDD");

    }

}

class B extends A{

    void read(){

        System.out.println("Read from USB");

    }

}

class C extends B{

    void read(){

        System.out.println("Read from Network");

    }

}

class Demo {

    public static void main(String[] args) {

        A ob1=new A();

        B ob2=new B();

        C ob3=new C();

        show(ob1);

        show(ob2);

        show(ob3);

    }

    static void show(A r){

        r.read();

    }

}

 //what is abstract class

 abstract class A{

     void show(){

         System.out.println("In A");

     }

    abstract void add(int x,int y);

 }

 class B extends A{

    @Override

    void add(int x, int y) {

        System.out.println(x+y);

    }

 }

class Demo2{

    public static void main(String[] args) {

         B ob=new B();

         ob.show();

         ob.add(100, 200);

    }

}

//why we need abstract class

//need of abstract class

abstract class Account{

    void getCustInfo(){

        System.out.println("Get cust info");

    }

    abstract void calculateInterest();

}

class Saving extends Account{

    @Override

    void calculateInterest() {

       System.out.println("Calculate interest of saving");

    }

}

class Loan extends Account{

    @Override

    void calculateInterest() {

        System.out.println("Calculate interest of Loan");

    }

}

public class Demo3 {

  public static void main(String[] args) {

    //create ref variable of abstract class

    Account r;

    r=new Saving();

    r.calculateInterest();

    r=new Loan();

    r.calculateInterest();

  }

}

//implementing multiple interface

interface A {

    void add(int x, int y);

}

interface B{

    public abstract void sub(int x, int y);

}

class Test implements A,B {

    @Override

    public void sub(int x, int y) {

      System.out.println(x+y);

    }

    @Override

    public void add(int x, int y) {

      System.out.println(x-y);

    }

}

class Demo4 {

    public static void main(String[] args) {

        Test t1 = new Test();

        t1.add(100, 200);

        t1.sub(90, 10);

    }

}

//implementing interface into class

interface A {

    void add(int x, int y);

    public abstract void sub(int x, int y);

}

class Test implements A {

    @Override

    public void sub(int x, int y) {

      System.out.println(x+y);

    }

    @Override

    public void add(int x, int y) {

      System.out.println(x-y);

    }

}

class Demo4 {

    public static void main(String[] args) {

        Test t1 = new Test();

        t1.add(100, 200);

        t1.sub(90, 10);

    }

}