LAB4:

1.CIRCLE

public class Circle {

private double radius;

private String color;

public Circle(double radius, String color) {

this.radius = radius;

this.color = color;

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

this.radius = radius;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

System.out.println("...." + color);

}

public double getArea() {

return (Math.PI \* this.radius \* this.radius);

}

public String toString() {

return "Circle [radius=" + radius + ", color=" + color + "]";

}

public class Cylinder extends Circle {

private double height;

public Cylinder(double radius, String color) {

super(radius, color);

this.height = height;

}

public double getHeight() {

return height;

}

public void setHeight(double height) {

this.height = height;

System.out.println("....");

}

public double getVolume() {

double a = getArea() \* this.height;

return a;

}

}

}

1. HinhHoc

package hinhhoc;

import java.util.Scanner;

public class hinhHoc {

public float PI = 3.14f;

public String ten;

public float chuVi;

public float dienTich;

public float theTich;

public hinhHoc() {

}

public void xuatTen() {

System.out.println("xuat ten" + ten);

}

public void inChuVi() {

System.out.println("in Chu Vi" + chuVi);

}

public void inDienTich() {

System.out.println("in Dien Tich" + dienTich);

}

public void inTheTich() {

System.out.println(" in The Tich" + theTich);

}

public class HinhTron extends hinhHoc {

public float banKinh;

public HinhTron(float banKinh) {

super();

this.banKinh = banKinh;

}

public HinhTron() {

}

public void nhapBanKinh() {

Scanner sc = new Scanner(System.in);

System.out.println("nhap ban kinh");

banKinh = sc.nextFloat();

}

public void nhapChuVi() {

Scanner sc = new Scanner(System.in);

chuVi = 2 \* PI \* banKinh;

System.out.println(" tinh chu vi" + chuVi);

}

public void nhapDienTich() {

Scanner sc = new Scanner(System.in);

dienTich = PI \* banKinh \* banKinh;

System.out.println(" tinh dien tich" + dienTich);

}

}

public class HinhChuNhat extends hinhHoc {

public float dai, rong;

public HinhChuNhat() {

}

public void nhapChieuDai() {

Scanner sc = new Scanner(System.in);

System.out.println("nhap chieu dai");

dai = sc.nextFloat();

}

public void nhapChieuRong() {

Scanner sc = new Scanner(System.in);

System.out.println(" nhap chieu rong");

rong = sc.nextFloat();

}

public void tinhChuVi() {

chuVi = (dai + rong) \* 2;

inChuVi();

}

public void tinhDienTich() {

dienTich = (dai \* rong);

inDienTich();

}

public class HinhTru extends HinhTron {

public float chieuCao;

public HinhTru(float banKinh, float chieuCao) {

super(banKinh);

this.chieuCao = chieuCao;

}

public HinhTru() {

}

public void nhapChieuCao() {

Scanner sc = new Scanner(System.in);

System.out.println("nhap chieu cao");

chieuCao = sc.nextFloat();

}

public void tinhTheTich() {

theTich = (PI \* chieuCao \* banKinh \* banKinh);

inTheTich();

}

public class HinhVuong extends HinhChuNhat {

public float canh;

public HinhVuong(float canh) {

super();

this.canh = canh;

}

public HinhVuong() {

}

public void nhapCanh() {

Scanner sc = new Scanner(System.in);

System.out.println("nhap canh");

canh = sc.nextFloat();

}

}

}

}

}

BTVN:

1.**Từ khóa super** là một biến tham chiếu được sử dụng để tham chiếu trực tiếp đến đối tượng của lớp cha gần nhất.

Từ khóa **super** có 3 cách sử dụng như sau:

1. Từ khóa super được sử dụng để tham chiếu trực tiếp đến biến instance của lớp cha gần nhất.
2. super() được sử dụng để gọi trực tiếp Constructor của lớp cha.
3. Từ khóa super được sử dụng để gọi trực tiếp phương thức của lớp cha.

VÍ DỤ:

class Vehicle {

    Vehicle() {

        System.out.println("Vehicle is created");

    }

}

class Bike2 extends Vehicle {

    Bike2() {

        super();//gọi Constructor của lớp cha

        System.out.println("Bike is created");

    }

    public static void main(String args[]) {

        Bike2 b = new Bike2();

    }

}

1. Kế thừa trong hàm khởi tạo:

Constructor trong java là một dạng đặc biệt của phương thức được sử dụng để khởi tạo các đối tượng.

Java Constructor được gọi tại thời điểm tạo đối tượng. Nó khởi tạo các giá trị để cung cấp dữ liệu cho các đối tượng, đó là lý do tại sao nó được gọi là constructor.

VÍ DỤ:

class Bike1 {

    Bike1() {

        System.out.println("Bike is created");

    }

    public static void main(String args[]) {

       Bike1 b=new Bike1();

    }

}

3.

1. ****Biến static (static variables):**** khi bạn khai báo một biến là static, thì biến đó được gọi là biến tĩnh, hay biến static.
2. ****Phương thức static (static methods):**** khi bạn khai báo một phương thức là static, thì phương thức đó gọi là phương thức static.
3. ****Khối static (static blocks):**** được sử dụng để khởi tạo thành viên dữ liệu static.
4. ****Import static****: từ phiên bản Java 5, cho phép import các thành viên tĩnh (static member) của một class hoặc package vào một class khác bằng cách sử dụng từ khóa import và sau đó sử dụng chúng như là thành viên của lớp đó.

4:Override phương thức

Override là hiện tượng 1 phương thức thuộc lớp cha được định nghĩa lại ở lớp con.

* Phương thức override và được override phải có chung kiểu trả về, tên phương thức và danh sách tham số.
* Override chỉ xảy ra giữa các lớp có quan hệ kế thừa.
* 1 phương thức ở lớp cha được override ở lớp con thì phương thức override ở lớp con không được phép thu hẹp tầm vực của phương thức đó ở lớp cha.
* Phương thức ở lớp cha được override ở lớp con thì phương thức override ở lớp con không được phép phát sinh những ngoại lệ kiểm tra (checked exception) khác loại hoặc không có quan hệ kế thừa với các ngoại lệ được ném ra từ phương thức đó ở lớp cha.
* Sử dụng từ khóa super để gọi phương thức được override ở lớp cha tại phương thức override ở lớp con.