**Câu 1: Hãy cho biết kết quả (và lý do) của đoạn chương trình sau:**

class Msg {  
 public Msg() {  
 System.*out*.println("Hello");  
 }  
 public void sayGoodbye() {  
 System.*out*.println("Goodbye");  
 }  
}  
class TestMsg extends Msg {  
 public TestMsg() {  
 super();  
 System.*out*.println("Hello Java");  
 }  
 public void sayGoodbye() {  
 System.*out*.println("Goodbye Java");  
 }  
}  
public class Demo {  
 public static void main(String args[]) {  
 Msg m1 = new Msg();  
 Msg m2 = new TestMsg();  
 m1.sayGoodbye();  
 m2.sayGoodbye();  
 }  
}

**Kết quả:**

Hello

Hello

Hello Java

Gooodbye Java

**Lời giải:**

Dòng 1: gọi đến hàm constructor không đối số Msg() và in ra: ‘Hello’

Dòng 2: Đây là upcasting (ép kiểu không tường minh), nó sẽ gọi đến constructor không đối số của TestMsg(), trong hàm này nó sẽ thực hiện gọi super() của hàm cha Msg() trước và in ra: ‘Hello’. Sau đó mới in ra ‘Hello Java’.

Dòng 3: thực hiện gọi tới hàm sayGoodbye() của Msg. và in ra : ‘Goodbye’.

Dòng 4: thực hiện gọi tới hàm sayGoodbye() của TestMsg và in ra : ‘Gooodbye Java’.

**Câu 2: Hãy cho biết kết quả (và lý do) của đoạn chương trình sau:**

class Msg {  
 public Msg() {  
 System.*out*.println("Hello");  
 }  
 public void sayGoodbye() {  
 System.*out*.println("Goodbye");  
 }  
}  
class TestMsg extends Msg {  
 public void sayGoodbye() {  
 System.*out*.println("Goodbye Java");  
 }  
}  
public class Demo {  
 public static void main(String args[]) {  
 Msg m1 = new Msg();  
 Msg m2 = new TestMsg();  
 m1.sayGoodbye();  
 m2.sayGoodbye();  
 }  
}

Kết quả:  
Hello

Hello

Goodbye

Goodbye Java

Lời giải:

Dòng 1: gọi hàm constructor không tham số Msg và in ra: ‘Hello’

Dòng 2: gọi đến hàm constructor TestMsg, trong hàm này sẽ gọi tự động đến hàm cha của nó và in ra: ‘Hello’.

Dòng 3: gọi đến hàm sayGoodbye của Msg và in ra: Goodbye

Dòng 4: gọi đến hàm sayGoodbye của TestMsg và in ra: Goodbye Java

Câu 3 Hãy cho biết kết quả (và lý do) của đoạn chương trình sau:

class Msg {  
 public Msg() {  
 }  
 public Msg(String str) {  
 System.*out*.println("Hello" + str);  
 }  
 public void sayGoodbye() {  
 System.*out*.println("Goodbye");  
 }  
}  
class TestMsg extends Msg {  
 public TestMsg(String str) {  
 System.*out*.println("Hello" + str);  
 }  
 public void sayGoodbye() {  
 System.*out*.println("Goodbye Java");  
 }  
}  
public class Demo {  
 public static void main(String args[]) {  
 Msg m1 = new Msg("Mark");  
 Msg m2 = new TestMsg("Java");  
 m1.sayGoodbye();  
 m2.sayGoodbye();  
 }  
}

kết quả:

HelloMark

HelloJava

Goodbye

Goodbye Java

Lời giải:

Dòng 1: gọi constructor có tham số của Msg và in ra: HelloMark

Dòng 2: gọi constructor có tham số của TestMsg, trong hàm này sẽ gọi constructor không tham số của hàm cha Msg trước rồi mới tới constructor của hàm con TestMsg sau và in ra: HelloJava

Dòng 3: gọi đến hàm sayGoodbye của Msg và in ra: Goodbye

Dòng 4: gọi đến hàm sayGoodbye của TestMsg và in ra: Goodbye Java

Câu 4 : Hãy chỉ ra lỗi sai của đoạn chương trình sau:

class Msg {  
 public Msg() {  
 System.*out*.println("Hello");  
 }  
 public void sayGoodbye() {  
 System.*out*.println("Goodbye");  
 }  
}  
class TestMsg extends Msg {  
 public void sayGoodbye() {  
 System.*out*.println("Goodbye Java");  
 }  
}  
public class Demo {  
 public static void main(String args[]) {  
 Msg m1 = new Msg("OOP");  
 TestMsg m2 = new Msg();  
 m1.sayGoodbye();  
 m2.sayGoodbye();  
 }  
}

Lỗi sau:

Dòng 1: sai vì gọi constructor có đối số trong khi Msg không có constructor có đối số truyền vào.

Dòng 2: dùng downcasting sai, vì hàm con TestMsg không thể là kiểu dữ liệu của hàm cha Msg được.

Câu 5 Hãy chỉ ra lỗi sai của đoạn chương trình sau:

interface Action{  
 public void drive(){};  
 public void stop(){};  
}  
class Vehicle implements Action{  
 private int id;  
 public void Vehicle(int id){  
 this.id = id;  
 System.*out*.println("Hello world");  
 }  
}  
public class Demo {  
 public static void main(String args[]){  
 Vehicle m = new Vehicle ();  
 }  
}

Lỗi sai:

+ Dòng 1: sai khi gọi hàm Vehicle không đối số .

+ trong interface Action: 2 hàm ảo thì không được có thân hàm

+ Vehicle implements Action: thì phải định nghĩa lại các hàm có trong interface Action đó.

Câu 6 Hãy chỉ ra lỗi sai của đoạn chương trình sau

interface Action{  
 public void drive();  
 public void stop();  
}  
class Vehicle implements Action{  
 private int id;  
 public void Vehicle(int id){  
 this.id = id;  
 System.*out*.println("Hello world");  
 }  
 public void drive(){  
 System.*out*.println("I’m driving");  
 }  
}  
public class Demo {  
 public static void main(String args[]){  
 Vehicle m = new Vehicle();  
 m.drive();  
 m.stop();  
 }  
}

Lỗi sai:

+ Vehicle không ghi đè lại 2 hàm ảo của interface Action.

+ gọi tới hàm Vehicle không đối số truyền vào.

Câu 7: Hãy cho biết kết quả (và lý do) của đoạn chương trình sau:

public class FileDemo{  
 public static void method1(File f1) {  
 String contents[] = f1.list();  
 for (int i=0; i< contents.length; i++){  
 System.*out*.println(contents[i] + "\n");  
 }  
 }  
 public static void main(String[] args){

*method1*(new File("D:\\Documents\\utt"));  
 }  
}

Kết quả:  
+ Đoạn code trên sẽ liệt kê toàn bộ tệp tin và thư mục có trong đường dẫn "D:\Documents\utt".

Giải thích:

+ new File("D:\\Documents\\utt")` được sử dụng để tạo một đối tượng kiểu File

+ phương thức `list()` được sử dụng để lấy danh sách các tệp tin và thư mục trong đường dẫn tương ứng với đối tượng kiểu `File`

Câu 9 Giải thích ý nghĩa và chỉ ra lỗi sai (giải thích) trong đoạn chương trình sau:

class MyForm extends Frame implements ActionListener{  
 public MyForm(String title){  
 super(title);  
 setSize(300, 200);  
 setVisible(true);  
 }  
}  
class ThiSinh{  
 private String name;  
 public ThiSinh(String name){  
 this.name = name;  
 }  
}  
class ThiSinhA extends ThiSinh{  
 double toan, ly, hoa;  
 public ThiSinhA(String name,double toan, double ly, double hoa){  
 super(name);  
 this.toan = toan;  
 this.ly = ly;  
 this.hoa = hoa;  
 }  
 public void display(){  
 System.*out*.println("Ho ten: " + this.name);  
 System.*out*.println("Tong diem: " + (this.toan + this.ly + this.hoa));  
 }  
}

Giải thích:

+ Lớp MyForm() : tạo cho chúng ta một cửa sổ (extends Frame) và triển khai giao diện ActionListener.

Xét kích thước setSize(300, 200) và setVisible(true) để cho nó hiển trị của sổ Frame lên.

+ Lớp ThiSinh là lớp cơ sở (lớp cha)

+ lớp ThiSinhA là lớp con kế thừa từ lớp cơ sở ThiSinh và triển khai các thuộc tính, phương thức

Lỗi sai:

+ this.name : thì không thể truy cập thuộc tính của lớp khác, vì thuộc tính đó ở dạng private

Câu 10: Cho đoạn chương trình sau:

Hãy giải thích ý nghĩa của try-catch trong chương trình. Cho biết kết quả nếu thực thi chương trình trên args[0] = “10” và trường hợp args[0] = “A10”.

public class Demo {  
 public static void main(String args[]) {  
 int value;  
 try {  
 value = Integer.*parseInt*(args[0]);  
 System.*out*.println(“value = “ + value);  
 }  
 catch(NumberFormatException ex) {  
 ex.printStackTrace();  
 }  
 }  
}

Giải thích ý nghĩa try-catch:

+ try-catch trong Java dùng để bắt lỗi (exception) .Nếu chương trình không phát hiện lỗi trong khối lệnh try thì nó sẽ chạy hết khối lệnh try và bỏ qua khối lệnh catch. Ngược lại nếu chương trình phát hiện có lỗi trong khối lệnh try thì sẽ dừng ngay lập tức và nhảy vào khối lệnh catch.

* Nếu args[0] = “10”. Thì chương trình trong khối lệnh try đúng và in ra: value = 10
* Nếu args[0] = “A10”. Thì chương trình trong khối lệnh try sai và nhảy ngay xuống catch và in ra lỗi đó.

Câu 11 Giả sử lớp Shape và Point được hiện thực như ở dưới. Phát biểu Point p = new Point(5,5); sẽ in ra màn hình nội dung gì? Giải thích.

class Shape {  
 public Shape() {  
 draw();  
 }  
 public void draw() {}  
}  
class Point extends Shape {  
 protected int x, y;  
 public Point(int xx, int yy) {  
 x = xx; y = yy;  
 }  
 public void draw() {  
 System.*out*.println("(" + x + "," + y + ")");  
 }  
}

Kết quả:

(0,0)

Giải thích:  
+ khi Point p = new Point(5,5); thì nó gọi constructor của lớp Point(int xx, int yy), tiếp đến nó gọi đến constructor của lớp cha Shape, constructor của lớp Shape thực thi và gọi đến hàm draw(), khi này hàm draw() được gọi đến là của lớp Point (vì nó được ghi đè) và in ra: (0, 0).

Câu 12 Chương trình dưới đây sẽ in ra màn hình nội dung gì? Giải thích.

class Base {  
 private void f() {

System.*out*.println("base f()");

}  
 public void show() {

f();

}  
}  
public class Derived extends Base {  
 public void f() {  
 System.*out*.println("derived f()");  
 }  
 public static void main(String[] args) {  
 Derived d = new Derived();  
 Base b = d;  
 b.show();  
 d.show();  
 }  
}

Kết quả:  
 base f()

base f()

Giải thích:

+ b.show() nó gọi tới hàm show() của lớp Base, mà vì show() không được ghi đè nên nó in ra: base f()

+ d.show() nó gọi tới hàm show() của lớp Base, mà vì show() không được ghi đè nên nó in ra: base f()

Câu 13 Giả sử lớp Person và phương thức main() được cài đặt như sau

biết nội dung gì sẽ được in ra màn hình. Giải thích.

class Person{  
 private String name;  
 public Person(String n){  
 name=n;  
 }  
 public void setName(String n){  
 name=n;  
 }  
 public void print(){  
 System.*out*.println(name);  
 }  
}  
  
public class Demo {  
 public static void main(String[] args){  
 Person p1 = new Person("Nemo");  
 Person p2 = p1;  
 p1.setName("Dory");  
 p1.print();  
 p2.print();  
 }  
}

Kết quả:  
 Dory

Dory

Giải thích:

+ vì Person p2 = p1 : nên cả 2 cùng trỏ tới 1 ô nhớ trong vùng nhớ Heap. Nền kết quả khi in ra đều giống nhau.

Câu 14 Chương trình dưới đây sẽ in ra màn hình nội dung gì? Giải thích.

class Base {  
 private void f() {

System.*out*.println("base f()");

}  
 public void show() {

f();

}  
}  
public class Derived extends Base {  
 public void f() {  
 System.*out*.println("derived f()");  
 }  
 public static void main(String[] args) {  
 Derived d = new Derived();  
 Base b = d;  
 b.show();  
 d.show();  
 }  
}

Kết quả:  
 base f()

base f()

Giải thích:

+ b.show() nó gọi tới hàm show() của lớp Base, mà vì show() không được ghi đè nên nó in ra: base f()

+ d.show() nó gọi tới hàm show() của lớp Base, mà vì show() không được ghi đè nên nó in ra: base f()

Câu 15: Giả sử lớp Person và phương thức main() được cài đặt như sau

class Person{  
 private String name;  
 public Person(String n){  
 name=n;  
 }  
 public void setName(String n){  
 name=n;  
 }  
 public void print(){  
 System.*out*.println(name);  
 }  
}  
  
public class Demo {  
 public static void main(String[] args){  
 Person p1 = new Person("Nemo");  
 Person p2 = p1;  
 p1.setName("Dory");  
 p1.print();  
 p2.print();  
 }  
}

Kết quả:  
 Dory

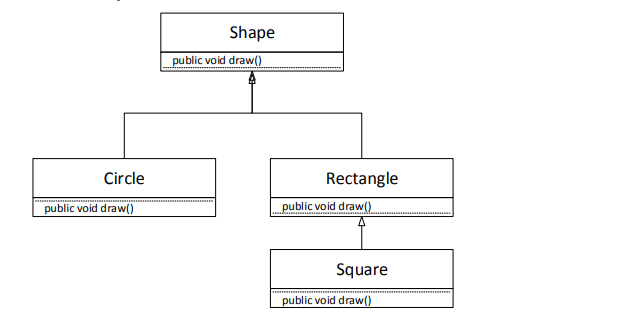
Dory

Giải thích:

+ vì Person p2 = p1 : nên cả 2 cùng trỏ tới 1 ô nhớ trong vùng nhớ Heap. Nền kết quả khi in ra đều giống nhau.

Câu 16: Cho cây kế thừa như hình vẽ:

Giả sử có phương thức boolean compare(Shape s1, Shape s2). Gọi compare(c1,c2) với c1, c2 là hai tham chiếu đến hai đối tượng thuộc lớp Circle có được hay không? Giải thích



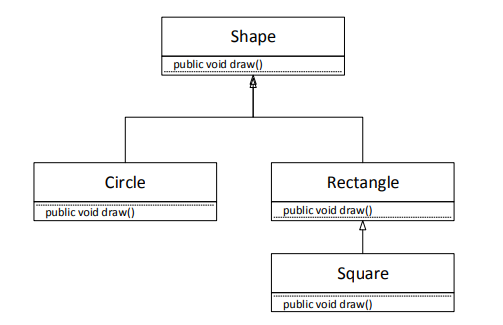
Trả lời: được

Giải thích:

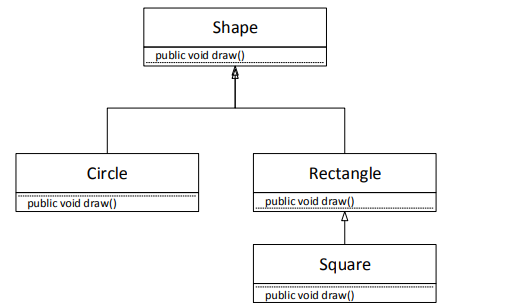
+ Circle kế thừa lớp Shape, vì vây trong trường hợp này Circle được hiểu là shape (upcasting- nhìn nhập lớp dẫn xuất như là lớp cơ sở).

Câu 17: Cho cây kế thừa như hình vẽ

Cho ví dụ về chuyển kiểu lên và chuyển kiểu xuống với các lớp ở cây thừa kế trên



Câu 18 : Cho cây kế thừa như hình vẽ:



Giả sử có phương thức show(Shape s) được cài đặt và sử dụng như ở dưới

void show(Shape s){  
 s.draw();  
}  
Shape c=new Circle();  
show(c);

Phương thức draw() được cài đặt trong lớp nào sẽ được gọi. Giải thích.

Câu 19 Giả sử lớp Shape và Point được hiện thực như ở dưới. Phát biểu Point p = new Point(5,5); sẽ in ra màn hình nội dung gì? Giải thích.

class Shape {  
 public Shape() {  
 draw();  
 }  
 public void draw() {}  
}  
class Point extends Shape {  
 protected int x, y;  
 public Point(int xx, int yy) {  
 x = xx; y = yy;  
 }  
 public void draw() {  
 System.*out*.println("(" + x + "," + y + ")");  
 }  
}

Câu 20 Định nghĩa 02 giao diện (interface) và 01 lớp cài đặt cả 2 giao diện đấy. Giải thích.

Câu 21 Cho ví dụ để phân biệt quan hệ is-a và has-a. Giải thích