Kiểm tra cuối kỳ Đề 1

Thời gian: 90 phút, *KHÔNG* được dùng tài liệu.

Câu 1. (2.5 điểm) Chỉ trả lời Đúng hoặc Sai.

a. Các lời gọi phương thức trong Java đều là liên kết động (Sai)

b. Nếu lớp con có phương thức trùng tên với một phương thức ở lớp cha, danh sách tham số của hai phương thức bắt buộc phải giống nhau (Sai)

c. Một lớp là trừu tượng thì bắt buộc phải chứa phương thức trừu tượng (Sai)

d. Phương thức khởi tạo của lớp cha sẽ được kế thừa ở lớp con (Sai)

e. Trong cùng một lớp, nếu hai phương thức có cùng tên thì bắt buộc phải khác tham số (Đúng)

f. Giao diện (interface) phải khai báo ít nhất một phương thức (Sai)

g. Trong Java, một lớp có thể có nhiều lớp cha và nhiều lớp con (Sai))

h. Nếu lớp A không định nghĩa bất cứ phương thức khởi tạo nào, A sẽ được trình biên dịch cung cấp phương thức khởi tạo mặc định (Đúng)

i. Nếu phương thức *String toString()* được cài đặt bởi một lớp nào đó, phương thức này bắt buộc phải là *public* (Đúng)

j. Phương thức thực thể có thể truy xuất các thuộc tính lớp (thuộc tính static) (Đúng)

Câu 2 (3 điểm)

a. Cơ chế chuyển kiểu lên (up casting) và chuyển kiểu xuống (down casting) là như thế nào, cho ví dụ minh họa? Khi nào không thể chuyển kiểu xuống được? (1 điểm)

Chuyển kiểu lên: nhìn nhận một đối tượng lớp con như là đối tượng lớp cha

Chuyển kiểu xuống: nhìn nhận đối tượng lớp cha như là đối tượng lớp con. Chuyển kiểu xuống chỉ thực hiện được khi đối tượng lớp cha vốn là đối tượng thuộc lớp con

Ví dụ:

class Person{}

class Employee extends Person{}

```
void test(){
Employee em=new Employee();
Person p=em; //up casting
Employee em2=(Employee)p;//down casting
}
```

b. Cho một ví dụ và giải thích về việc sử dụng giao diện (interface) làm kiểu của tham số của phương thức. (1 điểm)

```
public interface I1{};
void test(I1 o){...}
```

Trong ví dụ trên, phương thức test() sử dụng I1 làm kiểu của tham số. Lúc gọi test(), chúng ta cần truyền vào một đối tượng mà lớp của đối tượng đó có cài đặt giao diện I1.

c. Hãy định nghĩa một lớp ngoại lệ và cho một ví dụ thể hiện cách sử dụng lớp ngoại lệ đấy. (1 điểm)

```
class MyException extends Exception{
}
void test(){
        try{}
catch(MyException e){...}
}
```

Câu 3 (3 điểm)

Trong mô hình về sơ đồ (Diagram), một Diagram có thể chứa một hoặc nhiều hình. Hình (Shape) có thể là một trong các loại: Line (đường thẳng), Circle (đường tròn), Rectangle (hình chữ nhật). Dữ liệu về một Line bao gồm tọa độ điểm đầu và tọa độ điểm cuối trong hệ tọa độ hai chiều, về Circle là tọa độ điểm tâm đường tròn và bán kính, về Rectangle là tọa độ của

đỉnh trái trên và đỉnh phải dưới. Các đối tượng Shape cung cấp phương thức *print()* với nhiệm vụ in ra thông tin về hình vẽ.

a. Vẽ biểu đồ thiết kế lớp và viết mã cài đặt các lớp ở trên, sử dụng lớp trừu tượng và/hoặc giao diện (interface) một cách thích hợp. (1 điểm)
b. Trong lớp Diagram, định nghĩa phương thức *print()* in ra thông tin về tất cả các hình. (1 điểm)

```
public class Image{
    private ArrayList<Shape> shapes=new ArrayList<Shape>;
    public void print(){
    int n=shapes.size();
    for(int i=0;i<n;i++){
    shapes.get(i).print();
}
}
}
```

c. Giả sử trong một sơ đồ (Diagram) có thể chứa các sơ đồ khác. Hãy vẽ lại biểu đồ thiết kế lớp và định nghĩa lại phương thức *print()* cho lớp Diagram. (1 điểm)
Sinh viên có thể chuyển thành mẫu thiết kế Composite hoặc thêm thuộc tính Diagram[] children vào Diagram

Câu 4 (1.5 điểm)

a. Định nghĩa lớp tổng quát Triple và cho ví dụ cách sử dụng. Mỗi đối tượng thuộc lớp Triple có 3 thuộc tính *first*, *second*, và *third* thuộc cùng một lớp chưa biết trước. Lớp Triple có một phương thức khởi tạo có 3 tham số để khởi tạo cho các thuộc tính đấy. (1 điểm)

```
public class Triple<T>{
    T first;
T second;
T third;
public Triple(T f, T s, T t){
    first=f;
    second=s;
    third=t;
}
```

}

Cách dùng:

Triple<Integer> triple=new Triple(new Integer(1), new Integer(2), new Integer(3));

b. Mô tả mẫu thiết kế Singleton (0.5 điểm)

```
class Singleton{
    private static Singleton instance;
    private Singleton(){};
    public static Singleton getInstance(){
            if(instance==null)
                    instance=new Singleton();
            return instance;

        }
}
```
