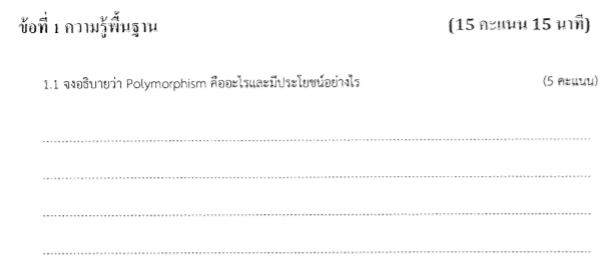
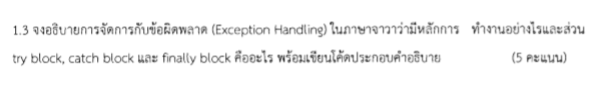
ข้อสอบ Final OOP 57



คลาสที่สืบทอดคลาสอื่นจะคงคุณสมบัติของ superclass ไว้ ทำให้สามารถสร้าง object ของคลาสนั้น ๆ โดยใช้ superclass เป็นตัวเก็บได้ ซึ่งช่วยให้สามารถเก็บ class หลาย ๆ class ใน object เดียวกันได้



เช่น การสร้าง class รูปทรงเรขาคณิต มีการคำนวณพื้นที่ ซึ่งรูปทรงเรขาคณิตแต่ละรูปมีสูตรคำนวณพื้นที่ที่แตกต่างกัน ดังนั้น ใน class รูปทรงเรขาคณิตที่มี method คำนวณพื้นที่จะยังไม่สามารถระบุได้ว่าใช้สูตรไหนในการคำนวณ ส่งผลให้ class รูปทรงเรขาคณิตต้องเป็น abstact class แล้วการสร้างคลาส รูปทรงสามเหลี่ยม หรือรูปทรงสี่เหลี่ยม ที่สืบทอดมาจาก class รูปทรงเรขาคณิตต้องระบุการคำนวณพื้นที่ของตัวมันเอง



การจัดการข้อผิดพลาดใช้ Exception ในการจัดการซึ่งมี 2 ประเภท คือ

1. Unchecked exception ที่จะตรวจสอบข้อผิดพลาดขณะรันโปรแกรม

2. Checked exception ที่ตรวจสอบขณะคอมไพล์ ซึ่งจำเป็นต้องใส่ทุกครั้ง เพื่อให้โปรแกรมคอมไพล์ผ่าน

Try

{

// try block สำหรับใส่การทำงานที่มีโอกาสเกิดข้อผิดพลาด

}

catch(Exception)

{

//เมื่อมีโค๊ดบรรทัดใดที่ผิดพลาดใน try block การทำงานใน try block จะหยุดแล้วเข้ามาใน catch block

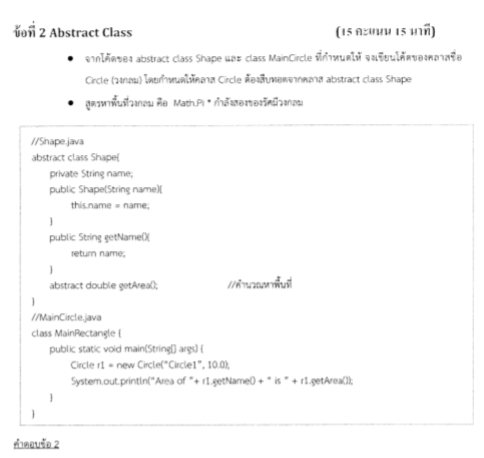
}

Finally

{

//หลังจากทำงานใน catch block เสร็จสิ้นจะเข้ามาใน finally block

}



public class Circle extends Shape

{

private double radius;

public Circle(String name, double radius)

{

super(name);

this.radius = radius;

}

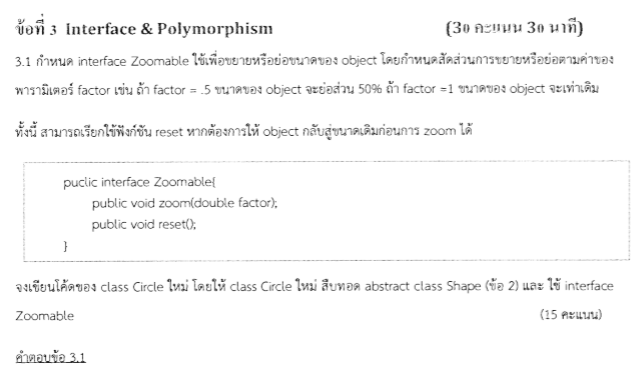
public double getArea()

{

return Math.PI\*radius\*radius;

}

}



public class NewCircle extends Shape implements Zoomable

{

private double radius;

private double factor;

public NewCircle(String name, double radius)

{

super(name);

this.radius = radius;

factor = 1;

}

public double getArea()

{

return Math.PI\*radius\*radius\*factor;

}

public void zoom(double factor)

{

this.factor = factor;

}

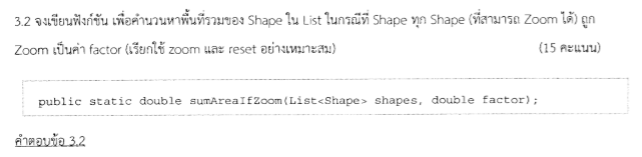
public void reset()

{

factor = 1;

}

}



public static double sumAreaIfZoom(List<Shape> shapes,double factor)

{

double sum = 0;

for(Shape s:shapes)

{

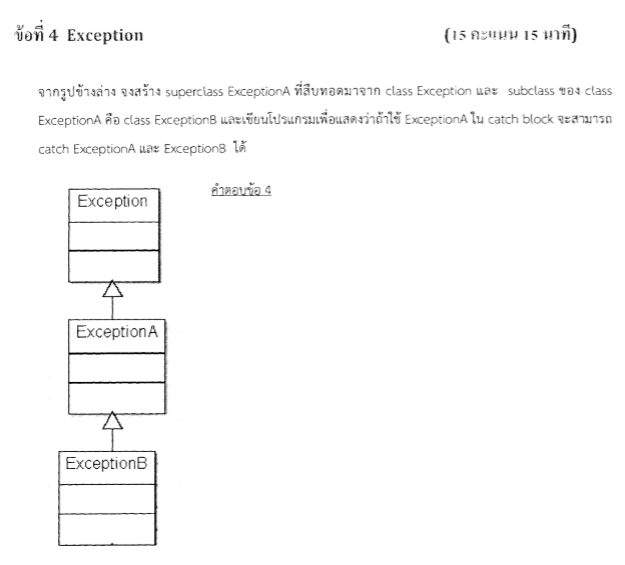
s.zoom(factor);

sum += s.getArea();

}

return sum;

}



Public class ExceptionA extends Exception

{

Public ExceptionA(){}

}

Public class ExceptionB extends ExceptionA

{

Public ExceptionB(){}

}

Public class Main

{

Public static void main(String[] args)

{

Try

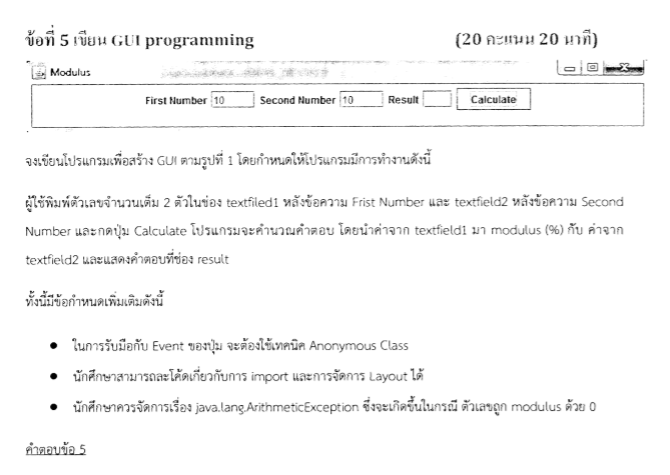
{}catch(ExceptionA ea)

{ea.printStackTrace();}

}

}

// ข้อนี้เดา



import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.util.\*;

public class ModulusGUI extends JFrame

{

private JLabel jlabel1;

private JLabel jlabel2;

private JTextField jtf1;

private JTextField jtf2;

private JLabel jlabelresult;

private JTextField jtfresult;

private JButton jbtn;

public ModulusGUI()

{

jlabel1 = new JLabel("First Number : ");

jtf1 = new JTextField(15);

jlabel2 = new JLabel("Second Number : ");

jtf2 = new JTextField(15);

jlabelresult = new JLabel("Result");

jtfresult = new JTextField(10);

jbtn = new JButton("Calculate");

jbtn.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent a)

{

try

{

int num1 = Integer.parseInt(jtf1.getText());

int num2 = Integer.parseInt(jtf2.getText());

jtfresult.setText(num1%num2+"");

}

catch(ArithmeticException e)

{

JOptionPane.showMessageDialog(null,"Error : Divide by zero");

}

}

});

Container c = getContentPane();

c.setLayout(new FlowLayout(FlowLayout.LEFT,5,5));

c.add(jlabel1);

c.add(jtf1);

c.add(jlabel2);

c.add(jtf2);

c.add(jlabelresult);

c.add(jtfresult);

c.add(jbtn);

this.setSize(900,300);

this.setLocationRelativeTo(null);

this.setDefaultCloseOperation(this.EXIT\_ON\_CLOSE);

this.setVisible(true);

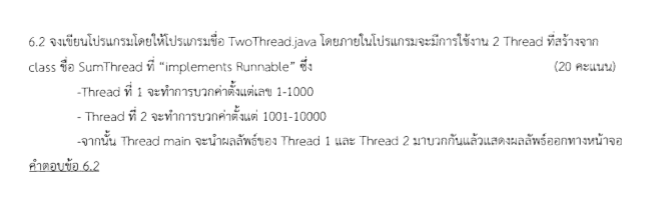
}

}



1. ทำการ extends Thread แล้วทำการ override เมธอด void run()

2. ทำการ implements Runnable แล้วทำการ override เมธอด void run()



import java.util.\*;

public class TwoThread

{

public static void main(String[] args)

{

SumThread s1 = new SumThread(1,1000);

SumThread s2 = new SumThread(1001,10000);

try

{

s1.start();

s2.start();

s1.join();

s2.join();

System.out.println("Sum = " + s1.getSum()+s2.getSum());

}catch(InterruptedException e){}

}

public static class SumThread extends Thread

{

private int start;

private int end;

private int sum;

public SumThread(int start,int end)

{

this.start = start;

this.end = end;

this.sum = 0;

}

public void run()

{

for(int i=start;i<=end;i++)

{

sum+=i;

}

}

public int getSum()

{

return sum;

}

}

}

\*\* ข้ออธิบายส่วนใหญ่อธิบายจากความเข้าใจ แนะนำให้หาคำตอบจาก google จะถูกต้องกว่า