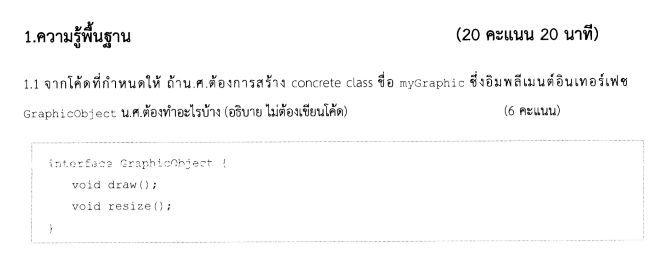
**ข้อสอบ Final OOP ปี 58**



1. ทำการสร้างคลาส ชื่อ myGraphic แล้วทำการ implements GraphicObject

2. เขียน Attributes และ Method ของคลาส

3. ทำการ Override (เขียนทับ) เมธอด void draw() และ เมธอด void resize()

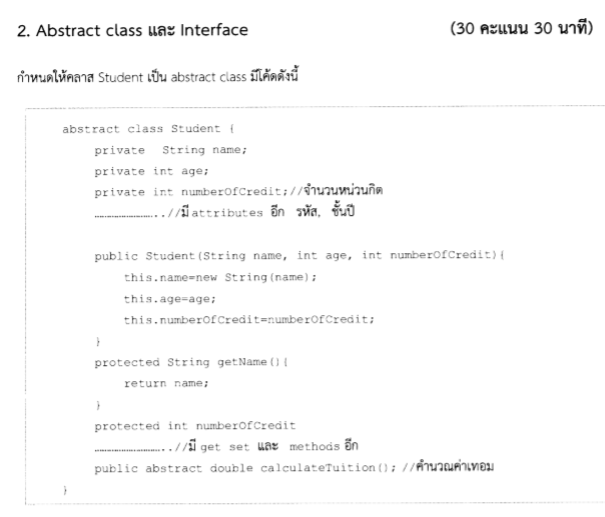


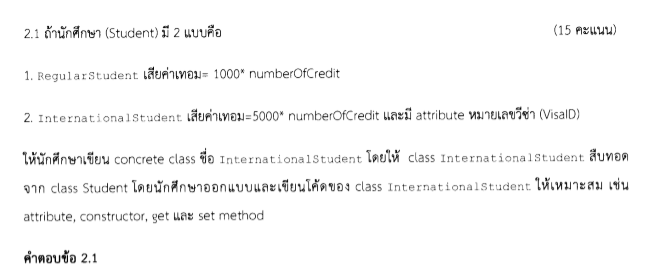
Thread คือ ส่วนการทำงานย่อยของโปรแกรม ซึ่งลำดับการทำงานจะแยกออกจากส่วนการทำงานหลัก ทำให้โปรแกรมสามารถทำงานแบบคู่ขนานได้ ซึ่งมีประโยชน์มาก ในกรณีที่มีงานที่ทำงานช้าอยู่ในโปรแกรม หากไม่นำส่วนนั้นแยกออกเป็น Thread แล้ว ก็จะทำให้การทำงานภาพรวมช้าไปด้วย



Checked exception คือ Exception ที่มีการตรวจสอบขณะคอมไพล์ หากไม่มีการเขียน try catch เพื่อจัดการ exception ประเภทนี้ โปรแกรมจะคอมไพล์ไม่ผ่าน

Unchecked Exception คือ Exception ที่มีการตรวจสอบขณะรันโปรแกรม หากไม่มีการเขียน try catch เพื่อจัดการ exception ประเภทนี้ โปรแกรมก็ยังคงสามารถคอมไพล์ผ่าน





public class InternationalStudent extends Student

{

private String visaID;

public InternationalStudent(String name, int age, int numberOfCredit,String visaID)

{

super(name,age,numberOfCredit);

this.visaID = visaID;

}

public String getVisaID()

{

return visaID;

}

public void setVisaId(String visaID)

{

this.visaID = visaID;

}

public double calculateTuition()

{

return this.getNumberOfCredit()\*1000;

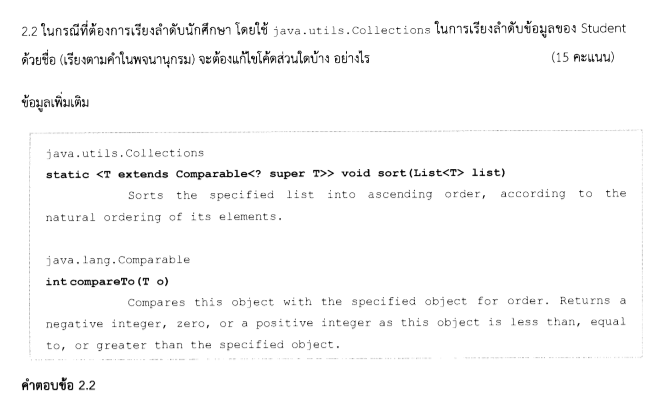
}

} Public InternationalStudent(String name, int age, int numberOfCredit, String visaID)

{

}

}



public abstract class Student implements Comparable<Student>

{

private String name;

private int age;

private int numberOfCredit;

public Student(String name, int age, int numberOfCredit)

{

this.name = name;

this.age = age;

this.numberOfCredit = numberOfCredit;

}

protected String getName()

{

return name;

}

protected int getNumberOfCredit()

{

return numberOfCredit;

}

public abstract double calculateTuition();

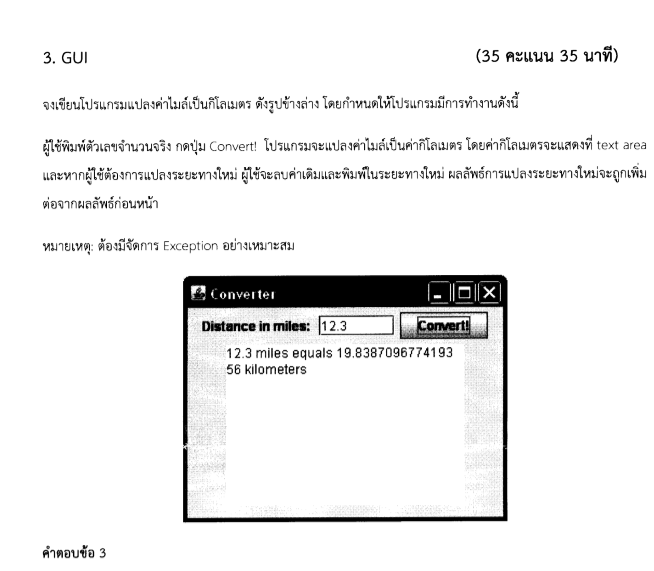
public int compareTo(Student st)

{

return name.compareTo(st.getName());

}

}



import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.util.\*;

public class DistanceConverter extends JFrame

{

private JLabel jlbl;

private JTextField jtf;

private JButton jbtn;

private JTextArea jta;

private JPanel jpn;

private String result;

public DistanceConverter()

{

super("Converter");

result = "";

jlbl = new JLabel("Distance in miles : ");

jtf = new JTextField(8);

jbtn = new JButton("Convert!");

jta = new JTextArea(20,20);

jbtn.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

String input = jtf.getText();

try

{

double mileValue = Double.parseDouble(input);

double kmValue = mileValue\*1.609344;

result = result + mileValue + " miles equals " + kmValue + " kilometers\n";

jta.setText(result);

}

catch(NumberFormatException nfe)

{

JOptionPane.showMessageDialog(null,"Input only number!");

}

}

});

jpn = new JPanel();

jpn.setLayout(new FlowLayout(FlowLayout.LEFT,2,5));

jpn.add(jlbl);

jpn.add(jtf);

jpn.add(jbtn);

Container c = getContentPane();

c.setLayout(new BorderLayout());

c.add(jpn,BorderLayout.NORTH);

c.add(jta,BorderLayout.CENTER);

this.setSize(350,280);

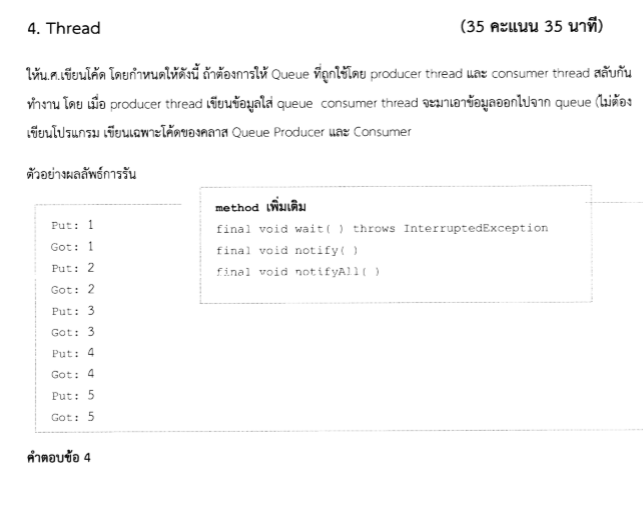
this.setLocationRelativeTo(null);

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

this.setVisible(true);

}

}



public static class Queue<DataType>

{

private LinkedList<DataType> l;

private boolean available;

public Queue()

{

l = new LinkedList<DataType>();

available = false;

}

public synchronized void put(DataType data)

{

while(available)

{

try

{

wait();

}catch(InterruptedException e){}

}

l.add(data);

available = true;

System.out.println("Put : " + data);

notifyAll();

}

public synchronized DataType get()

{

while(!available)

{

try

{

wait();

}catch(InterruptedException e){}

}

DataType data = l.removeFirst();

available = false;

System.out.println("Got : " + data);

notifyAll();

return data;

}

}

public static class Producer implements Runnable

{

Queue<Integer> q;

public Producer(Queue<Integer> q)

{

this.q = q;

}

public void run()

{

for(int i = 0;i<10;i++)

{

q.put(i);

try

{

Thread.sleep(1000);

}catch(InterruptedException ie){}

}

}

}

public static class Consumer implements Runnable

{

Queue<Integer> q;

public Consumer(Queue<Integer> q)

{

this.q = q;

}

public void run()

{

while(true)

{

q.get();

}

}

}

Code : <http://github.com/tanakorn0314/OOP_final>