**20161795 고 가해**

**abstract class**

The abstract class is inheritance relationships with java, one inheritance relationship just can be used with one class. There is clear method overwriting require at the subclass what abstract class inheritance. But classic class can choose is it should be overwriting. The abstract class have more abstract method than classic class. The other part is same as classic class and classic class object can be directly instantiation but abstract class object must during Up casting to achieve it.

**Interface**

The interface is similar to abstract class. But class just can inheritance with one class. Interface can inheritance muliti-inheritance. abstract class and interface have different design idea. abstract class is expression "is-a” relationship but interface is expression "like-a” relationship.

Code 1 analyzes

abstract class Calculator {  
 //define 3 abstract method  
 public abstract int add(int a, int b);  
  
 public abstract int subtract(int a, int b);  
  
 public abstract double average(int[] a);  
  
}

public class GoodCalc extends Calculator {  
  
 // to the GoodClac has been defined is not abstract class,so if should relized abstract superclass’s all abstract method.  
  
 public int add(int a, int b) {  
 return a + b;  
 }  
  
 public int subtract(int a, int b) {  
 return a - b;  
 }  
  
 public double average(int[] a) {  
 double sum = 0;  
 for (int i = 0; i < a.length; i++)  
 sum += a[i];  
 return sum / a.length;  
 }  
  
 public static void main(String[] args) {  
 Calculator c = new GoodCalc();  
 System.*out*.println(c.add(2, 3));  
 System.*out*.println(c.subtract(2, 3));  
 System.*out*.println(c.average(new int[]{2, 3, 4}));  
 }  
  
}

Code 2 analyzes

//define the interface  
interface PhoneInterface {  
 int *BUTTONS* = 20;  
  
 void sendCall();  
  
 void receiveCall();  
}  
  
//inheritance the phoneInterface ‘s interface and define 2 interface method  
interface MobilePhoneInterface extends PhoneInterface {  
 void sendSMS();  
  
 void receiveSMS();  
}  
  
//define the interface  
interface MP3Interface {  
 public void play();  
  
 public void stop();  
}  
  
//define the class  
class PDA {  
 //define the PDA method

public int calculate(int x, int y) {  
 return x + y;  
 }  
}  
  
//SmartPhone class inheritance PDA，and realized MobilePhoneInterface、MP3Interface interface  
class SmartPhone extends PDA implements MobilePhoneInterface, MP3Interface {  
 public void sendCall() {  
 System.*out*.println("Send Call");  
 }  
  
 public void receiveCall() {  
 System.*out*.println("Recevie Call");  
 }  
  
 public void sendSMS() {  
 System.*out*.println("Send SMS");  
 }  
  
 public void receiveSMS() {  
 System.*out*.println("Receive SMS");  
 }  
  
 public void play() {  
 System.*out*.println("Play");  
 }  
  
 public void stop() {  
 System.*out*.println("Stop");  
 }  
  
 public void schedule() {  
 System.*out*.println("Schedule");  
 }  
}  
  
public class InterfaceEx {  
 public static void main(String[] args) {  
 SmartPhone p = new SmartPhone();  
 p.sendCall();  
 p.play();  
 System.*out*.println(p.calculate(3, 5));  
 p.schedule();  
 }  
}

**Exception Handling**

There’s many exceptions in java, there’s abnormal condition make program exit. we must be handing exception for avoid program exit. some exception at java, throw user-defined exception/ When code have exception like IOException, ArithmeticException, SocketException

Exception Handling keyword

Try block/ catch block / finally block/ throw block / throws blocks / Try-with-Resources

The java exception inheritance relationship

Code 1 Analyze

import java.util.Scanner;  
public class ExceptionExam1 {  
 public static void main (String[] args) {  
 Scanner rd = new Scanner(System.*in*);  
 int divisor = 0;  
 int dividend = 0;  
  
 System.*out*.print("Input dividend:");  
 dividend = rd.nextInt();  
 System.*out*.print("Input divisor:");  
 divisor = rd.nextInt();  
 System.*out*.println(dividend+" / "+divisor+" = "+dividend/divisor);  
 System.*out*.println("Exception 1");  
 }  
}

if the divisor is 0 it will be having exception java.lang.ArithmeticException

import java.util.Scanner;  
public class ExceptionExam1 {  
 public static void main (String[] args) {  
 Scanner rd = new Scanner(System.*in*);  
 int divisor = 0;  
 int dividend = 0;  
  
 System.*out*.print("Input dividend:");  
 dividend = rd.nextInt();  
 System.*out*.print("Input divisor:");  
 divisor = rd.nextInt();  
 try{  
 System.*out*.println(dividend+" / "+divisor+" = "+dividend/divisor);  
 }catch(java.lang.ArithmeticException ex ) {  
  
  
 System.*out*.println("Exception 1");  
 }  
 }  
}

Code 2 Analyze

public class ExceptionExam2 {  
 public static void main (String[] args) {  
 int[] intArray = new int[5];  
 intArray[0] = 0;  
 for (int i = 0; i < 5; i++) {  
 intArray[i+1] = i+1 + intArray[i];  
 System.*out*.println("intArray[" + i + "]" + " = " + intArray[i]);  
 }  
 System.*out*.println("Exception 2");  
 }  
}

This code exception is we have 5 array but i<5 when i=4 the under code i+4=5 so out of bands we can catch the java.lang.ArrayIndexOutOfBoundsException

public class ExceptionExam2 {  
 public static void main(String[] args) {  
 int[] intArray = new int[5];  
 intArray[0] = 0;  
 for (int i = 0; i < 5; i++) {  
 try {  
 intArray[i + 1] = i + 1 + intArray[i];  
 System.*out*.println("intArray[" + i + "]" + " = " + intArray[i]);  
 } catch (java.lang.ArrayIndexOutOfBoundsException ex) {  
 System.*out*.println("Exception 2");  
 }  
 }  
 }  
}

Code 3 Analyze

public class ExceptionExam3{  
 public static void main (String[] args) {  
 String[] stringNumber = {"23", "12", "998", "3.141592"};  
 for (int i = 0; i < stringNumber.length; i++) {  
 int j = Integer.*parseInt*(stringNumber[i]);  
 System.*out*.println("stringNumber[" + i + "]" + " = " + j);  
 }  
 System.*out*.println("Exception 3");  
 }  
}

int can’t defined number out of integer so there’s java.lang.NumberFormatException come out

public class ExceptionExam3 {  
 public static void main(String[] args) {  
 String[] stringNumber = {"23", "12", "998", "3.141592"};  
 for (int i = 0; i < stringNumber.length; i++) {  
 try {  
 int j = Integer.*parseInt*(stringNumber[i]);  
 System.*out*.println("stringNumber[" + i + "]" + " = " + j);  
 } catch (java.lang.NumberFormatException ex) {  
 System.*out*.println("Exception 3");  
 }  
 }  
 }  
}