Question 1

Write the exception class **WrongCourseException**that would produce the following output:

|  |
| --- |
| **Code** |
| **public class TestCourse{**  **public static void courseNumber(intnum) throws WrongCourseException{**  **if (num != 111){**  **throw new WrongCourseException("This course is not CSE 111");**  **}**  **}**  **public static void main(String[] args){**  **try{**  **courseNumber(110);**  **}catch(WrongCourseException e){**  **System.out.println("ERROR: "+e);**  **}**  **}**  **}** |
| **Output** |
| **ERROR: This course is not CSE 111** |

Question 2

**Write the output of the following code:**

**public class ParentException extends Exception{**

**protected String msg;**

**public ParentException(Object o){**

**msg = o.toString();**

**}**

**public String toString() {**

**return "Parent: "+ msg;**

**}**

**}**

**public class ChildException extends ParentException{**

**publicChildException(Object o){**

**super(o);**

**}**

**public String toString() {**

**return "Child: "+ msg;**

**}**

**}**

**public class BadThing{**

**private String msg = null;**

**privateintnum;**

**publicBadThing (int i){**

**num = i;**

**}**

**public void badMethod() throws Exception{**

**if (num< 11){**

**throw new ArrayIndexOutOfBoundsException ();**

**}**

**if (num%4 == 0){**

**throw new ParentException(new BadThing(num));**

**}else{**

**throw new ChildException(new BadThing(num));**

**}**

**}**

**public String toString(){**

**if (num%4 == 0){**

**return "You are an divisible "+ num;**

**}else{**

**return "You are an not divisible "+ num;**

**}**

**}**

**}**

**public class GoodThing{**

**public static void goodMethod(Object o) throws Exception{**

**try{**

**((BadThing)o).badMethod();**

**}catch(ChildException c){**

**System.out.println("goodThing: "+c);**

**}catch(Exception e){**

**throw(e);**

**}**

**}**

**public static void main(String [] args){**

**int i = 0;**

**for (i =7; i <95; i+=9){**

**try{**

**goodMethod(new BadThing(i));**

**}catch(ParentException p){**

**System.out.println("main: "+p);**

**}catch(Exception e){**

**System.out.println("Boo Hoo! I could not stop it ");**

**}**

**}**

**}**

**}**

**[Answer on the question paper]**

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Question 3

**Write the output of the following code:**

**public class ParentException extends Exception{**

**protected String msg;**

**publicParentException(Object o){**

**msg = o.toString();**

**}**

**public String toString() {**

**return "Parent: "+ msg;**

**}**

**}**

**public class ChildException extends ParentException{**

**publicChildException(Object o){**

**super(o);**

**}**

**public String toString() {**

**return "Child: "+ msg;**

**}**

**}**

**public class BadThing{**

**private String msg = null;**

**privateintnum;**

**publicBadThing (int i){**

**num = i;**

**}**

**public void badMethod() throws Exception{**

**if (num< 11){**

**throw new ArrayIndexOutOfBoundsException ();**

**}**

**if (num%2 == 0){**

**throw new ParentException(new BadThing(num));**

**}else{**

**throw new ChildException(new BadThing(num));**

**}**

**}**

**public String toString(){**

**if (num%2 == 0){**

**return "Even "+ num;**

**}else{**

**return "Odd "+ num;**

**}**

**}**

**}**

**public class GoodThing{**

**public static void goodMethod(Object o) throws Exception{**

**try{**

**((BadThing)o).badMethod();**

**}catch(ParentException p){**

**System.out.println("goodThing: "+p);**

**throw p;**

**}catch(Exception e){**

**throw(e);**

**}**

**}**

**public static void main(String [] args){**

**int i = 0;**

**for (i =8; i < 24; i+=3){**

**try{**

**goodMethod(new BadThing(i));**

**}catch(ChildException p){**

**System.out.println("main: "+p);**

**}catch(RuntimeException e){**

**System.out.println("Main RuntimeException");**

**}catch(Exception e){**

**System.out.println("Main Exception");**

**}**

**}**

**}**

**}**

|  |
| --- |
| **OUTPUT** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| **public class MyException extends Exception{** |
| **private String myStr;** |
| **public MyException(String init){** |
| **myStr = init;** |
| **}** |
| **public String toString(){** |
| **return "My Error: "+ myStr + 2 + 3;** |
| **}** |
| **}** |
| **public class RTExcept {** |
|  |
| **public static void whyNot() throws MyException {** |
| **throw new MyException("225");** |
| **}** |
|  |
| **public static void throwit() {** |
| **try{** |
| **System.out.println("throwit ");** |
| **whyNot();** |
|  |
| **}catch(Exception e){** |
| **System.out.println(e);** |
| **throw new RuntimeException();** |
| **}finally{** |
| **System.out.println("Thrown the exception");** |
| **}** |
| **}** |
|  |
| **public static void main(String[] args) {** |
| **try {** |
| **System.out.println("hello ");** |
| **throwit();** |
| **} catch (Exception re) {** |
| **System.out.println("caught ");** |
| **} finally {** |
| **System.out.println("finally ");** |
| **}** |
| **System.out.println("after ");** |
| **}** |
| **}** |

What is the output of the code above?

|  |  |
| --- | --- |
| 1. | 6. |
| 2. | 7. |
| 3. | 8. |
| 4. | 9. |
| 5. | 10. |

Given the following classes, write the code for the **BBAStudent**class so that the following output is printed when we run the **TestStudent**class.

**Name : Default BBA Student**

**Name : Barack Obama**

**Name : Joe Biden**

public abstract class Student{

private String name;

public final String getName(){

return name;

}

public final void setName(String name){

this.name = name;

}

}

public class TestStudent{

public static void printName(Student s){

System.out.println("Name : "+s.getName());

}

public static void main(String [] args){

printName(new BBAStudent());

printName(new BBAStudent("Barack Obama"));

printName(new BBAStudent("Joe Biden"));

}

}

#### **(Answer on the answer script)**

Write the **Mango** and the **Jackfruit** classes so that the following code generates the output below:

|  |
| --- |
| **public class Test{** |
| **public static void testFruit(Fruit f){** |
| **System.out.println("----Printing Detail-----");** |
| **if(f.hasFormalin()){** |
| **System.out.println("Do not eat the "+f.getName()+".");** |
| **System.out.println(f);** |
| **}else{** |
| **System.out.println("Eat the "+f.getName()+".");** |
| **System.out.println(f);** |
| **}** |
| **}** |
| **public static void main(String [] args){** |
| **Mango m = new Mango();** |
| **testFruit(m);** |
| **Jackfruit j = new Jackfruit();** |
| **testFruit(j);** |
| **}** |
| **}** |
| **public class Fruit{** |
| **private boolean formalin = false;** |
| **public String name = "";** |
| **public Fruit(boolean formalin, String name){** |
| **this.formalin = formalin;** |
| **this.name = name;** |
| **}** |
| **public String getName(){** |
| **return name;** |
| **}** |
| **public booleanhasFormalin(){** |
| **return formalin;** |
| **}** |
| **}** |

**----Printing Detail-----**

**Do not eat the Mango.**

**Mangos are bad for you**

**----Printing Detail-----**

**Eat the Jackfruit.**

**Jackfruits are good for you**

|  |
| --- |
| **public class Frodo extends Bilbo** |
| **{** |
| **public void method1()** |
| **{** |
| **System.out.println("Frodo 1");** |
| **super.method1();** |
| **}** |
|  |
| **public void method3()** |
| **{** |
| **System.out.println("Frodo 3");** |
| **}** |
| **}** |
| **public class Gandalf** |
| **{** |
| **public void method1()** |
| **{** |
| **System.out.println("Gandalf 1");** |
| **}** |
|  |
| **public void method2()** |
| **{** |
| **System.out.println("Gandalf 2");** |
| **method1();** |
| **}** |
| **}** |
| **public class Bilbo extends Gandalf** |
| **{** |
| **public void method1()** |
| **{** |
| **System.out.println("Bilbo 1");** |
| **}** |
| **}** |
| **public class Gollum extends Gandalf** |
| **{** |
| **public void method3()** |
| **{** |
| **System.out.println("Gollum 3");** |
| **}** |
| **}** |

**And assuming the following variables have been defined:**

**Gandalf var1 = new Frodo();**

**Gandalf var2 = new Bilbo();**

**Gandalf var3 = new Gandalf();**

**Object var4 = new Bilbo();**

**Bilbo var5 = new Frodo();**

**Object var6 = new Gollum();**

In the table below, indicate in the right-hand column the output produced by

the statement in the left-hand column. If the statement produces more than one

line of output, indicate the line breaks with slashes as in "a/b/c" to indicate

three lines of output with "a" followed by "b" followed by "c". If the

statement causes an error, fill in the right-hand column with either the phrase

"compiler error" or "runtime error" to indicate when the error would be

detected.

**Statement Output**

|  |  |
| --- | --- |
| **var1.method1();** |  |
| **var2.method1();** |  |
| **var4.method1();** |  |
| **var6.method1();** |  |
| **var1.method2();** |  |
| **var3.method2();** |  |
| **var4.method2();** |  |
| **var5.method2();** |  |
| **var6.method2();** |  |
| **((Frodo)var4).method3();** |  |
| **((Frodo)var6).method2();** |  |
| **((Gollum)var1).method3();** |  |
| **((Gollum)var4).method1();** |  |
| **((Gandalf)var1).method2();** |  |
| **((Frodo)var4).method1();** |  |
| **((Gollum)var6).method2();** |  |
| **((Gandalf)var2).method1();** |  |
| **((Bilbo)var6).method2();** |  |
| **((Frodo)var1).method3();** |  |
| **((Gandalf)var5).method3();** |  |