**Exercise 1.**

**1. What is the output of the following? 4 Points**

|  |
| --- |
| public class ExcepTest{ |
|  |
| public static void main(String args[]){ |
| int a[] = new int[2]; |
| try{ |
| System.out.println("Access element three :" + a[3]); |
| }catch(ArrayIndexOutOfBoundsException e){ |
| System.out.println("Exception thrown"); |
| } |
| finally{ |
| a[0] = 6; |
| System.out.println("First element value: " +a[0]); |
| System.out.println("The finally statement is executed"); |
| } |
| } |
| } |

Exception thrown **…………2**

First element value: 6 **…………1**

The finally statement is executed **…………1**

**2. What will happen when you attempt to compile and run the following class? 4 Points**

|  |
| --- |
| public class Base{ |
| public Base(int i){ |
| System.out.println("Base"); |
| } |
| } |
|  |
| public class Second extends Base { |
| public static void main(String args[]){ |
| Second s = new Second (); |
| } |
| public Second (){ |
| System.out.println("Second"); |
| } |
| } |

Choose the right answer

1. Compilation and output of the string "Second" at runtime
2. Compilation and no output at runtime
3. Compilation and output of the string "Base"
4. Compile time error: An error occurs at the constructor of the class Second because this constructor calls implicitly the default constructor (without parameter constructor) of the class Base which does not exist.

**EXERCISE 2**

Give the output of the following program. **10 Points**

**public** **class** Flight {

**private** String flightNum;

**protected** **int** dist;

**public** Flight() {

flightNum = "Unknown";

dist = 500;

System.*out*.println ("The Flight is Created");

}

**public** Flight (String flightNum, **int** dist) {

**this**.flightNum = flightNum;

t**his**.dist=dist;

}

**public** **void** display() {

System.*out*.println ("Flight number: " + flightNum + " distance: " + dist );

}

**public** **int** cost () **throws** Exception {

**return** 200;

}

}

**public** **class** LongDistanceFlight **extends** Flight {

**protected** **int** rate;

**public** LongDistanceFlight () {

rate = 2;

}

**public** LongDistanceFlight (String flightNum, **int** dist, **int** r) {

**super**(flightNum, dist);

rate = r;

}

**public** **void** display () {

System.*out*.println ("Long Distance Flight ");

**super**.display();

}

**public** **int** cost() **throws** Exception {

**if** (dist < 1000 ) **throw** **new** Exception (

"Exception: Distance Less Than 1000 Km");

**return** (**super**.cost()+ dist\*rate);

}

}

**public** **class** InternationalFlight **extends** LongDistanceFlight{

**protected** **int** airportFee;

**public** InternationalFlight(String s, **int** d, **int** r, **int** f) {

**super**(s,d,r);

airportFee = f;

}

**public** InternationalFlight(**int** f) {

airportFee = f;

}

**public** **void** display() {

System.*out*.println ("International Flight ");

**super**.display();

**try** {

System.*out*.println(**this**.cost());

}

**catch** (Exception e) {System.*out*.println(e.getMessage());

}

}

**public** **int** cost() **throws** Exception {

**return** (**super**.cost()+airportFee);

}

}

**public** **class** TestFlights {

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

**int** i;

Flight [] flightList = **new** Flight[2];

flightList[0] = **new** InternationalFlight("SV3875", 1000, 3, 100);

flightList[1] = **new** InternationalFlight(500);

**for** (i=0; i< 3; i++) {

System.*out*.println("Iteration " + (i+1));

**try** {

flightList[i].display();

} **catch** (Exception e) { System.*out*.println(

"Exception in Iteration " + (i+1));

}

}// end for

}// end main

}

The Flight is Created **…………1**

Iteration 1

International Flight **…………1**

Long Distance Flight **…………1**

Flight number: SV3875 distance: 1000 **…………1**

3300 **…………1**

Iteration 2

International Flight **…………1**

Long Distance Flight **…………1**

Flight number: Unknown distance: 500 **…………1**

Exception: Distance Less Than 1000 Km **…………1**

Iteration 3

Exception in Iteration 3 **…………1**

**EXERCISE 3**



***HumanResource interface:***

* + METHODS:
    - ***display()***: this method displays all the attributes.
    - ***calculateSalary***: calculates and returns the salary.

***Employee*** class

* + METHODS:
    - ***Employee(id: int, name: string, basicSalary: double)***: constructor.
    - ***display()***: this method displays all the attributes of the employee.
    - ***calculateSalary***: returns the salary of the employee which is calculated as following:
      * *For* ***Faculty***:

salary = basic salary + (**number of extra hours** \* extra hours rate \* 4) + (number of years of experience \* 500).

* + - * *For* ***Secretary***:

salary = basic salary + (rank \* 2000).

***Faculty*** class

* + Attributes:
    - ***position***: the position of the Faculty.
    - ***effectiveWorkingHours***: the number of working hours.
    - ***extraHoursRate***: the rate of an extra hour.
    - ***expectedWorkingHours:*** The minimum load of the faculty.
    - ***nbYearsOfExperience:*** The number of years of experience.
  + METHODS:
    - ***Faculty(id: int, name: string, basicSalary: double, expectedWorkingHours: int)***: constructor.
    - ***display()***: this method displays the salary and all the attributes of the Faculty.
    - ***getNbExtraHours***: this method returns the number of extra hours which is calculated as following:

*number of extra hours = effective working hours - expected working hours.*

***N.B:***

* *If the number of extra hours is less than 0 this method throws an exception.*
* the number of extra hours is obtained **only** using the method *getNbExtraHours.*

***Secretary*** class

* + Attributes:
    - ***rank***: the rank of the Secretary.
  + METHODS:
    - ***Secretary(id: int, name: string, basicSalary: double, rank: int)***: constructor.
    - ***display()***: this method displays the salary and all the attributes of the secretary.

**QUESTION:** Translate into Java code the interface HumanResource, the class ***Employee*** and the class ***Faculty*** .

**public** **interface** HumanResource { **…………1**

**public** **void** display(); **…………1**

**public** **double** calculateSalary(); **…………1**

}

**public** **abstract** **class** Employee **implements** HumanResource { **…………1 + …………1**

**private** **int** id; **…………0.5**

**private** String name; **…………0.5**

**protected** **double** basicSalary; **…………0.5**

**public** Employee(**int** i, String s, **double** d) {

id = i; **…………1**

name = s; **…………1**

basicSalary = d; **…………1**

}

**public** **void** display() {

System.*out*.println(id + name + basicSalary); **…………3**

}

}

**public** **class** Faculty **extends** Employee { **…………1**

**private** String position; **…………0.5**

**private** **int** effectiveWorkingHours; **…………0.5**

**private** **double** extraHoursRate; **…………0.5**

**private** **int** expectedWorkingHours; **…………0.5**

**private** **int** nbYearsOfExperience; **…………0.5**

**public** Faculty(**int** i, String s, **double** bs, String pos, **int** exwh, **int** years) {

**super**(i, s, bs); **…………2**

position = pos; **…………1**

expectedWorkingHours = exwh; **…………1**

nbYearsOfExperience = years; **…………1**

}

**public** **void** display() {

**super**.display(); **…………2**

System.*out*.println(position + effectiveWorkingHours+

extraHoursRate + expectedWorkingHours +

nbYearsOfExperience); **…………2.5**

System.*out*.println(calculateSalary());**…………2.5**

}

**public** **int** getNbExtraHours() **throws** Exception{ **…………1**

**int** nbExtra = effectiveWorkingHours - expectedWorkingHours; **…………2**

**if** (nbExtra < 0) **…………1**

**throw** **new** Exception("Number of Extra Hours Less than 0"); **…………2**

**return** nbExtra; **…………1**

}

**public** **double** calculateSalary () {

**double** salary = 0.0 ;

**try** { **…………1**

salary = basicSalary +

(getNbExtraHours() \* extraHoursRate \*4) +

(nbYearsOfExperience \* 500.0); **…………3**

} **catch** (Exception e) { System.out.println(e.getMessage()); } **………1 + ……1**

**return** salary; **…………1**

}

}