**Java Activity 2**

**Inheritance**

**Question 1.1**

**The class NumConversion is already available that performs number conversion**

|  |
| --- |
| **NumConversion** |
| int num1 |
| **convert2Oct()**  **convert2Bin()**  **convert2Hexa()** |

**Convert2Oct() 🡪 converts the given num1 to octal**

**Convert2Bin() 🡪 converts the given num1 to binary**

**Convert2Hexa() 🡪 converts the given num1 to hexa decimal**

**Client requires the new class StdCalculator in OO way that contains the behavior**

**Convert2Oct() 🡪 converts the given num1 to octal**

**Convert2Dec() 🡪 converts the given num1 to decimal**

**Convert2Hexa() 🡪 converts the given num1 to hexa decimal**

**Sum() -> sum of two numbers**

**Mul() -> multiply of two numbers**

**Sub() 🡪 subtract two numbers**

**Div()🡪 divide two integer number**

**Mod()🡪find remainder**

**Use appropriate OO concepts**

**Question 1.2**

The existing class Factorial contains doFact() 🡪 find factorial for the state x

The new class Factorian is also Factorial have to perform

findFactorian() for the state num🡪 145 = 1! + 4! + 5!

|  |
| --- |
| **Factorial** |
| Int x; |
| doFact() |

|  |
| --- |
| **Factorian** |
| num |
| findFactorian() |

Use appropriate OO concepts

**Question 2.1**

The existing class displays all the items in an array in ascending order and searches the given string in an array using linear search

|  |
| --- |
| **Searcher** |
| **-String[] arr //hardcoded sorted values** |
| **void display()**  **int findItem(String str)// linear search** |

**The new class contains the behavior**

**display() 🡪 displays an array in ascending order**

**findItem(String str) 🡪 searches the element using binary search**

**findItem(int idx)🡪 returns the element for given idx**

**Question 2.2**

The existing class palindrome checks the state and displays the given string is palindrome or not

|  |
| --- |
| **palindrome** |
| String str |
| checkPalindrome() |

The new class PalindromeList contains the state String[] and checks the palindrome for the given string and also for the String array.

Use appropriate concepts

**Question 2.3**

The existing class SapientList

* displays all the items in an array in respective of size
* returns the size(no of elements added into an array)
* adds element to the array , the string must match 3 – 10 alphabets
* removes last element from the array

|  |
| --- |
| **SapientList** |
| **-String[] arr = new String[100]**  **-Int size** |
| **getSize()**  **add(String str)**  **void display()**  **int getItem(int idx)**  **remove()** |

**The new class SapientNewList contains all the behavior of SapientList and**

* **override display() to display in ascending order**
* **remove(idx) -> removes the element at specific index**

**RTP**

**Question 3.1**

**There are 2 types of product**

**Computer**

|  |  |
| --- | --- |
| **state** | **productId, productName, price, hdd** |
| **behaviour** | **display()🡪 displays computer details**  **IsExpensive() 🡪 returns expensive if price >= 50000**  **Returns average if price >= 30000 & < 50000**  **Else return cheap** |

**Mobile**

|  |  |
| --- | --- |
| **state** | **productId, productName, price,camera** |
| **behaviour** | **display()🡪 displays mobile details**  **IsExpensive() 🡪 returns expensive if price >= 10000**  **Returns average if price >= 5000 & < 10000**  **Else return cheap** |

**Bind the computer and mobile objects in to an array**

**Display the product report having productId, productName, price, range and product type**

**Question 3.2**

**There are 2 types of Employee**

**OnrollEmp**

|  |  |
| --- | --- |
| **state** | **empId, empName, sal, da, pf** |
| **behaviour** | **display()🡪 displays onroll employee details**  **calcSal() 🡪 the computation is sal + da -pf** |

**ContractEmp**

|  |  |
| --- | --- |
| **state** | **empId, empName, sal, days** |
| **behaviour** | **display()🡪 displays contract employee details**  **calcSal() 🡪 the computation is sal \* days** |

**Bind the onroll and contract employee objects in to an array**

**Display the salary report having emptId, empName, sal, da, pf, days, net salary and employee type**

**Question 3.3**

**There are 2 types of employee in the company**

**Trainee**

|  |  |
| --- | --- |
| **state** | **empId, empName, percentage** |
| **behaviour** | **display()🡪 displays trainee details**  **isoutstanding() 🡪 if percentage >=90** |

**Professor**

|  |  |
| --- | --- |
| **state** | **empId, empName, rating** |
| **behaviour** | **display()🡪 displays professor details**  **isoutstanding() 🡪 if rating >= 5** |

Bind the Trainee and Professor objects in to an array

Display the outstanding report having emptId, empName, rating, percentage, Isoutstanding and employee type

**Question 3.4**

Client maintains 2 types of account in the bank

**Savings**

|  |  |
| --- | --- |
| **state** | **accId, accName, bal** |
| **behaviour** | **display()🡪 displays Savings details**  **deposit(amt) 🡪 amount is added to bal field** |

**Loan**

|  |  |
| --- | --- |
| **state** | **accId, accName, bal** |
| **behaviour** | **display()🡪 displays loan account details**  **deposit(amt) 🡪 amt is withdrawn from the balance** |

The two type of account is hardcoded and maintained at static level in the client app.

Client app prompts the user to select transactions either savings or loan account and enters the amount.

It process the respective account and displays the respective account details

The client must prompt the user to continue

**Interface(RTP)**

**Question 5.1**

Create the shape interface that contains calcArea()

Circle, Rectangle and Triangle objects implements Shape interface

Shape[] can contain Circle, Rectangle , Triangle objects

Display shape type, states , area of respective shape

**Question 5.2**

Create the Shape interface that contains calcVolume()

Cylinder, Cone and Sphere objects implements Shape interface

Shape[] can contain Cylinder, Cone and Sphere objects

Display shape type, states , volume of respective shape

**Question 5.3**

**The class Calculator having methods**

|  |  |
| --- | --- |
| **Convert2Oct()** |  |
| **Convert2Dec()** |  |
| **Convert2Hexa()** |  |
| **findLeap()** |  |
| **findFact()** |  |
| **findQuadratic()** |  |

* **All the methods are not given to the client to access. Only required methods must be given to access**
* **The Calculator class have to be accessed by two type of clients**
* **Conversion** 🡪 **One of the client requires only conversion methods**
* **Scientific** 🡪 **the other client requires only findLeap(), findFact(), findQuadratic()**
* **Use appropriate OO design and concepts**