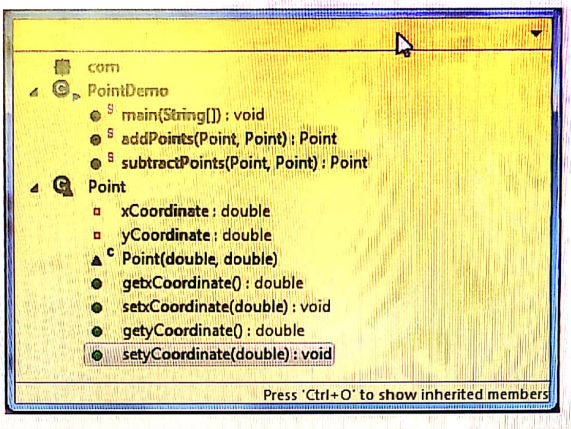
Create class structure as mentioned in the outline within com package.

addPoints and subtractPoints methods are static and would return Point object. First parameter in both methods is left operand and second parameter is right operand.

Test the same creating required objects in main method.



Close

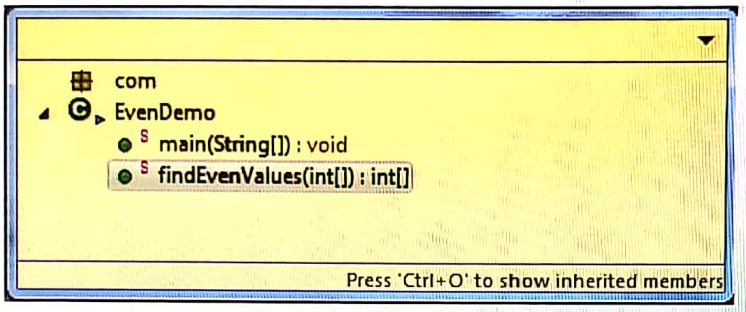
Scanned by CamScanner

Create class EvenDemo in com package with main method.

Add one more static method – findEvenValues which will take input as int array and return int array which will have all even numbers identified from input array.

E.g input array: 22,11,9,20,13

Output array: 22,20



Close

Scanned by CamScanner

Scanned by CamScanner

After Question 2 is completed, add one more method in EngineInventory class and one customized exception as per below outline:

```
com
EngineInventoryDemo
    main(String[]) : void
EngineInventory
    engines : ArrayList<Engine>
    EngineInventory()
    addEngine(Engine) : int
    searchEngine(char) : int
    enginesWithMileageGreaterThan(double) : int
NoEngineExistException
    getMessage() : String
```

enginesWithMileageGreaterThan – This method will take a mileage value as parameter. It will return the count of engines in the array list "engines" which has mileage greater than the input parameter. However, if there is no such engine where mileage is greater than input parameter, then the method will throw NoEngineExistsException object.

NoEngineExistsException – This class extends Exception and overrides getMessage() method by returning below String (case sensitive):

"No Engine Exists With Greater Mileage"

Scanned by CamScanner

will throw NoEngineExistsException object.

NoEngineExistsException – This class extends Exception and overrides getMessage() method by returning below String (case sensitive):

"No Engine Exists With Greater Mileage"

Try below code from main method and verify the output:

Input:

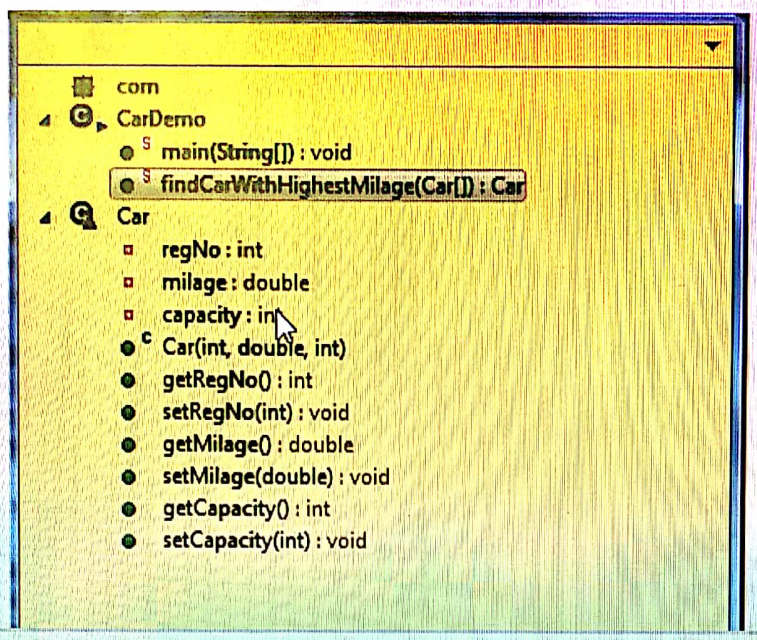
```
EngineInventory inv = new EngineInventory();

Engine e1 = new TwoStroke(1, 2200, 25, 'd');
Engine e2 = new FourStroke(2, 2400, 25, 'g');
Engine e3 = new TwoStroke(3, 2600, 25, 'd');
Engine e4 = new FourStroke(4, 2800, 25, 'd');
System.out.println(inv.addEngine(e1));
System.out.println(inv.addEngine(e2));
System.out.println(inv.addEngine(e3));
System.out.println(inv.addEngine(e4));
try {
    System.out.println(inv.enginesWithMileageGreaterThan(20));
} catch (NoEngineExistException e) {
    // TODO Auto-generated catch block
    System.out.println(e.getMessage());
}
try {
    System.out.println(inv.enginesWithMileageGreaterThan(50));
} catch (NoEngineExistException e) {
    // TODO Auto-generated catch block
    System.out.println(e.getMessage());
}
```

Close

Scanned by CamScanner

Create class structure in com package as displayed in the outline.

The method findCarWithHighestMilage will return reference to Car object with highest mileage in given array. Assume that no two cars have same mileage.



Close

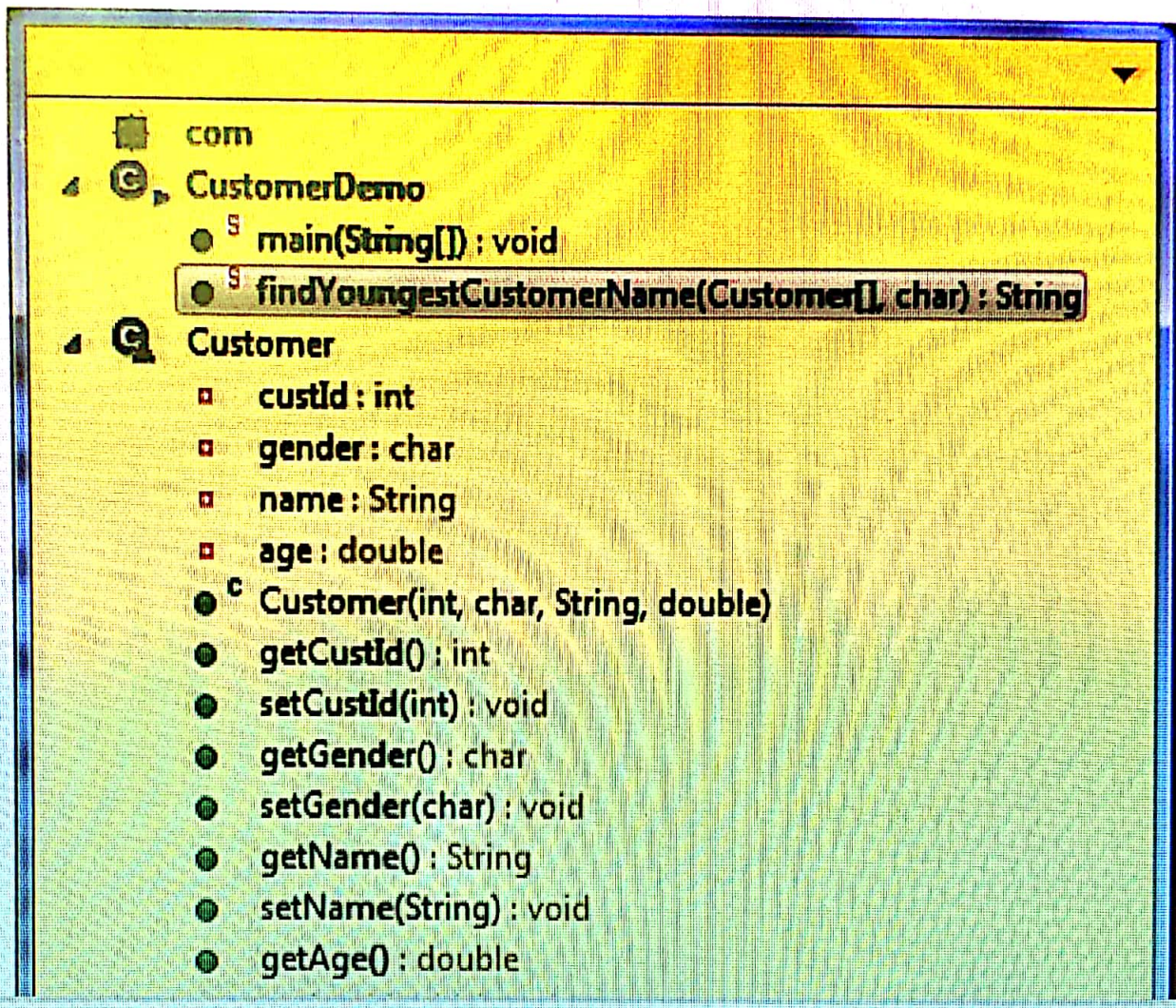
Scanned by CamScanner

Create class structure in com package as per the outline.

The method – findYoungestCustomerName will take the Customer ar

It will return name of Customer who is youngest.

Assume that there are no two customers with same age.



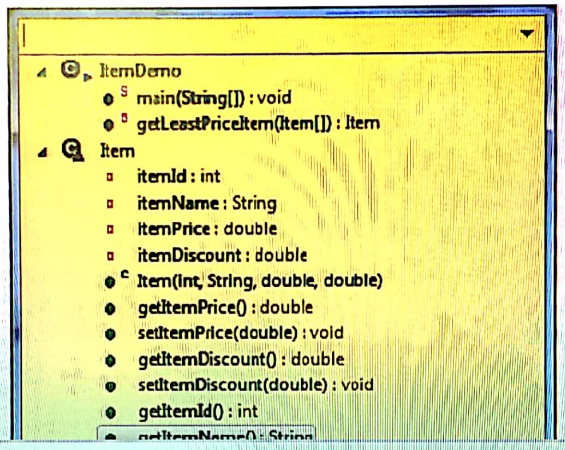
Scanned by CamScanner

Question

Create class Item with below attributes:
itemId
itemName
itemPrice
itemDiscount

Create class ItemDemo with main method. Declare array of 5 item objects in main method. Initialize this array. Declare another method in this class – getLeastPriceItem which will take item array as input and return item with least price (considering discount as well). Display item name in main method. Follow class outline diagram as given below. Ensure class attributes are private and other methods are public.

Use package "com" to build your solution.



Scanned by CamScanner

Create package com.

Create classes as per below outline:

- com
  - EngineDemo
    - main(String[]) : void
  - Engine
    - id : int
    - volume : double
    - fuelCapacity : double
    - fuelType : char
    - getMileage() : double
    - Engine(int, double, double, char)
    - getId() : int
    - setId(int) : void
    - getVolume() : double
    - setVolume(double) : void
    - getFuelCapacity() : double
    - setFuelCapacity(double) : void
    - getFuelType() : char
    - setFuelType(char) : void
  - TwoStroke
    - TwoStroke(int, double, double, char)

Abstract method

Parameter sequence: id, volume, fuelCapacity, fuelType

Close

Scanned by CamScanner

Question

Fuel type value should be either 'g' or 'd' ('g' indicating that engine runs on gas and 'd' indicates that engine runs on diesel. Validation is not required. Just specify value as 'd' or 'g').

Mileage calculation formula:

Two Stroke Engine:

Gas:

Mileage = volume*2*3/1000 + fuelCapacity/2

Diesel:

Mileage = volume*2*2/1000 + fuelCapacity/4

Four Stroke Engine:

Gas:

Mileage = volume*4*3/1000 + fuelCapacity/2

Diesel:

Mileage = volume*4*2/1000 + fuelCapacity/4

Close

Scanned by CamScanner

Question

Try below code from main method and verify the output:

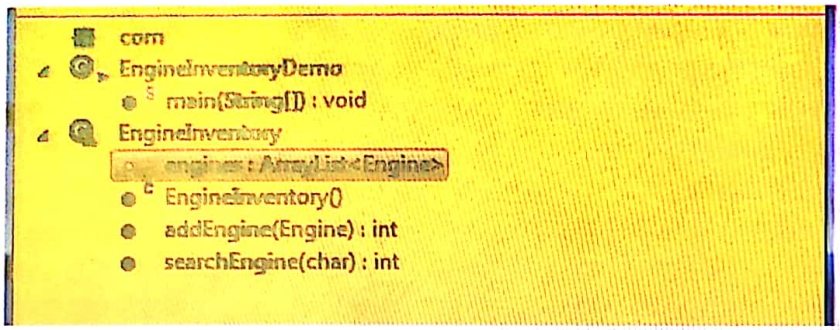
```
public static void main(String[] args) {
    // TODO Auto-generated method stub

    Engine e1 = new TwoStroke(1, 2000, 25, 'g');
    Engine e2 = new FourStroke(2, 2000, 25, 'g');
    Engine e3 = new TwoStroke(3, 2000, 25, 'd');
    Engine e4 = new FourStroke(4, 2000, 25, 'd');

    System.out.println(e1.getMileage());
    System.out.println(e2.getMileage());
    System.out.println(e3.getMileage());
    System.out.println(e4.getMileage());
```

Output:

```
Console
<terminated> En
24.5
36.5
14.25
22.25
```

Scanned by CamScanner

After Question 1 is completed, create below classes in the same com package:



addEngine method will add the Engine object to the engines array list and return the current size of that list. However, it will return -1 if engine with specified id already exists. If specified Engine object is null, then also return -1.

searchEngine will return the engines count based on specified fuelType as parameter ('d' or 'g').

Test your code with below input in main method and specified output:

Input:

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

    EngineInventory inv = new EngineInventory();
```

Scanned by CamScanner

Input:

```
public static void main(String[] [redacted]) {

    EngineInventory inv = new EngineInventory();

    Engine e1 = new Twostroke(1, 2000, 25, 'g');
    Engine e2 = new Fourstroke(2, 2000, 25, 'g');
    Engine e3 = new Twostroke(3, 2000, 25, 'd');
    Engine e4 = new Fourstroke(2, 2000, 25, 'd');

    System.out.println(inv.addEngine(e1));
    System.out.println(inv.addEngine(e2));
    System.out.println(inv.addEngine(e3));
    System.out.println(inv.addEngine(e4));

    System.out.println(inv.searchEngine('g'));
    System.out.println(inv.searchEngine('d'));

}
```

Output:

```
1
2
3
-1
2
1
```

Once done, submit the solution files over iAscert.

Close

Scanned by CamScanner

Question

After Question 2 is completed, add one more method in EngineInventory class and one customized exception as per below outline:

```
com
  EngineInventoryDemo
    main(String[]) : void
  EngineInventory
    engines : ArrayList<Engine>
    EngineInventory()
    addEngine(Engine) : int
    searchEngine(char) : int
    enginesWithMileageGreaterThan(double) : int
  NoEngineExistException
    getMessage() : String
```

enginesWithMileageGreaterThan – This method will take a mileage value as parameter. It will return the count of engines in the array list "engines" which has mileage greater than the input parameter. However, if there is no such engine where mileage is greater than input parameter, then the method will throw NoEngineExistsException object.

NoEngineExistsException – This class extends Exception and overrides getMessage() method by returning below String (case sensitive):

```
"No Engine Exists With Greater Mileage"
```

Try below code from main method and verify the output:

Close

Scanned by CamScanner

returning below String (case sensitive):

"No Engine Exists With Greater Mileage"

Try below code from main method and verify the output:

Input:

```
EngineInventory inv = new EngineInventory();

Engine e1 = new TwoStroke(1, 2200, 25, 'd');
Engine e2 = new FourStroke(2, 2400, 25, 'g');
Engine e3 = new TwoStroke(3, 2600, 25, 'd');
Engine e4 = new FourStroke(4, 2800, 25, 'd');
System.out.println(inv.addEngine(e1));
System.out.println(inv.addEngine(e2));
System.out.println(inv.addEngine(e3));
System.out.println(inv.addEngine(e4));
try {
    System.out.println(inv.enginesWithMileageGreaterThan(20));
} catch (NoEngineExistException e) {
    // TODO Auto-generated catch block
    System.out.println(e.getMessage());
}
try {
    System.out.println(inv.enginesWithMileageGreaterThan(50));
} catch (NoEngineExistException e) {
    // TODO Auto-generated catch block
    System.out.println(e.getMessage());
}
```

Close

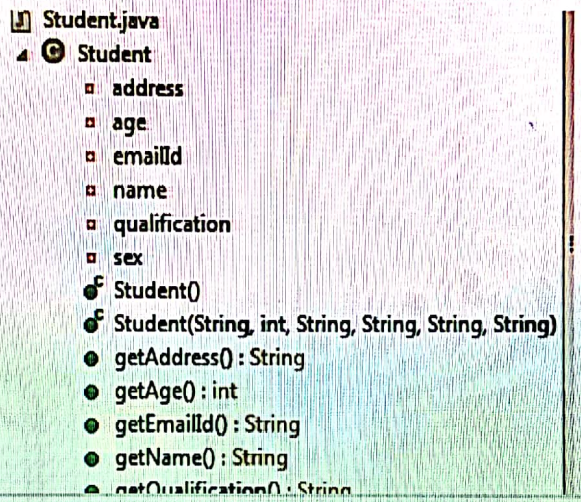
Scanned by CamScanner

Question

name
age
sex
address
emailId
qualification

Provide getters ,setters for all the attributes.
Provide constructors(default,parameterized)constructors for initializing student object

Follow the below Class outline diagram



Scanned by CamScanner

Question

- getQualification() : String
- getSex() : String
- setAddress(String) : void
- setAge(int) : void
- setEmailId(String) : void
- setName(String) : void
- setQualification(String) : void
- setSex(String) : void

Create a class AgeException which is a custom exception with a default constructor.Below is the class outline diagram

AgeException.java
- AgeException
  - AgeException()

Create a class University with the following methods:

1)String[] addStudent(Student[] stuObj)

Close

Scanned by CamScanner

Question

This method adds collection of students to the university .

If a student is successfully added it should store a message in return array as

Student with the following details is added successfully to university

otherwise it should store the below message in return array

Student cannot be added to university as age is greater than 35 and less than 23

public void registerStudent(Student studentObj)throws AgeException

This method has a student parameter object (containing all the current student details) – This method validates all the student details and throws AgeException if the student age is >35 or <23.

Below is the class outline for University.

- University.java
  - University
    - addStudent(Student[]) : String[]
    - registerStudent(Student) : void

Close

Scanned by CamScanner

Question

Create all classes in package com (lower case).
Please avoid any usage of System.out.println in the program.
Create class PizzaKing with below outline:
In main method, call the method performPizzaKingOperations.

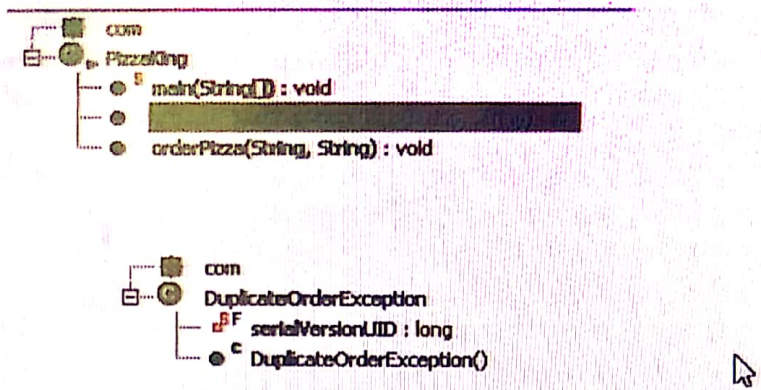
1. orderPizza : This method will compare both orders, if they are same, even with different cases, then the method will throw a DuplicateOrderException
2. performPizzaKingOperations : This method takes parameters of pizza name that we want to order. For example Tomato , Capsicum, etc. Call orderPizza method from here.
   Exceptions should be handled in performPizzaKingOperations method only. This method should also be able to handle any other type of exceptions which are not known or declared.
   The performPizzaKingOperations method will execute for following scenarios depending on the 2 inputs provided and return an integer value as the result of execution.

Everything works successfully for order names, for ex. Onion and Capsicum, then return 0
If both orders are same/duplicate , for example paneer and Paneer, and the DuplicateException was encountered then return -1
If any unknown exception occurs, then the result should be returned as -2.
Before exiting this method in any case, a result should be returned.

- com
  - PizzaKing
    - main(String[]) : void
    - performPizzaKingOperations(String, String) : int
    - orderPizza(String, String) : void

Close

Scanned by CamScanner



Close

Scanned by CamScanner

unique names sorted in ascending order of the names. Note duplicate names if any should be removed.

Assumption Name of Student will always will be in CAPITAL LETTER.

Follow class outline diagram as given below. Ensure class attributes are private and other methods are public. Note there should not be any System.out.println statements in any of the methods in Student class and SchoolClerk class.

Use package "com" to build your solution

com

Student

- id : int
- name : String
- age : byte
- city : String
- getId() : int
- setId(int) : void
- getName() : String
- setName(String) : void
- getAge() : byte
- setAge(byte) : void
- getCity() : String
- setCity(String) : void

Close

Scanned by CamScanner

setAge(byte) : void
getCity() : String
setCity(String) : void

com
import declarations
SchoolClerk
getUniqueStudentNames(List<Student>) : TreeSet<String>

Close

Scanned by CamScanner

Question

EmployeeDemo

Employee

- empId : int
- name : String
- basicPay : double
- perksPay : double
- Employee(int, String, double, double)
- getName() : String
- setName(String) : void
- getBasicPay() : double
- setBasicPay(double) : void
- getPerksPay() : double
- setPerksPay(double) : void
- getEmpId() : int
- calculateSalary() : double

Associate

- Associate(int, String, double, double)
- calculateSalary() : double

Consultant

- consultingFees : double
- getConsultingFees() : double
- setConsultingFees(double) : void
- Consultant(int, String, double, double, double)
- calculateSalary() : double

Create child class Associate and implement calculateSalary method which will return sum of basic pay

Close

Scanned by CamScanner

Question

and perks pay.

Create child class Consultant which also has one attribute – consultingFees. This amount is also added to basic pay and perks pay to calculate salary in implementation of the abstract method. Constructor of Consultant class will take consultingFees (as last parameter) along with other parameters.

Test this code in main method with below input and outputs:

```
package com;

public class EmployeeDemo {

    public static void main(String[] args) {

        Employee e1 = new Associate(1, "aaa", 5000, 7000);
        Employee e2 = new Consultant(2, "bbb", 3000, 1000, 5000); //last parameter is consulting fees
```

Scanned by CamScanner

```
package oop

public class EmployeeDemo {

    public static void main(String[] args) {

        Employee [illegible] = new Associate(1, "aaa", 5000, 7000);
        Employee e2 = new Consultant(2, "bbb", 3000, 1000, 5000); //last parameter is consulting fees
        System.out.println([illegible].calculateSalary());
        System.out.println([illegible].calculateSalary());
    }
}
```

Console

```
<terminated> EmployeeDemo [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (Mar 2, 2015 5:43:52 PM)
12000.0
9000.0
```

Close

Scanned by CamScanner

Question

For the question solved before, add below method in Organization class and test the same with sample input/output shared.

sortEmployees: This method will return tree set of Employee objects by sorting the objects based on employee id.

Also, add the BasicPayException class as per described.

Outline:

```
com
EmployeeDemo
    main(String[]) : void
BasicPayException
    basicSalary : double
    BasicPayException(double)
    getMessage() : String
Organization
    employees : ArrayList<Employee>
    Organization()
    sortEmployees() : TreeSet<Employee>
    addEmployee(Employee) : int
    searchEmployee(String) : int
    getEmployeeIdWithMaxSalary() : int
```

Close

Scanned by CamScanner

My Progress Meter: 13.39 /100

Question

BasicPayException class is extended from Exception. It has one attribute – basicSalary which is set through constructor and returned in getMessage() method.

This exception is thrown in addEmployees method when the employee being added has basicPay < 0.

Sample Input:

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

    Organization o = new Organization();

    try
    {
    System.out.println(o.addEmployee(new Associate(3, "aaa", 5000, 7000)));
    System.out.println(o.addEmployee(new Associate(1, "aaa", 4000, 6000)));
    System.out.println(o.addEmployee(new Consultant(2, "bbb1", 4000, 1000, 5000)));
    System.out.println(o.addEmployee(new Consultant(1, "bbb2", 3000, 1000, 5000)));
    System.out.println(o.addEmployee(new Consultant(4, "bbb3", 3000, 8000, 2000)));
    System.out.println(o.addEmployee(new Consultant(5, "bbb3", -3000, 8000, 2000)));
    }
    catch(BasicPayException e)
    {
        System.out.println(e.getMessage());
    }

    System.out.println(o.searchEmployee("aaa"));
    System.out.println(o.searchEmployee("bbb"));
    System.out.println("id with max salary: " + o.getEmployeeIdWithMaxSalary());
    System.out.println("ids:");
    TreeSet<Employee> set = o.sortEmployees();
    for(Employee e: set) System.out.println(e.getEmpId());

}
```

Close

Scanned by CamScanner

Output:

```
<terminated> EmployeeDemo [Java
1
2
3
-1
4
-3000.0
2
0
id with max salary: 4
ids:
1
2
3
4
```

Close

Scanned by CamScanner

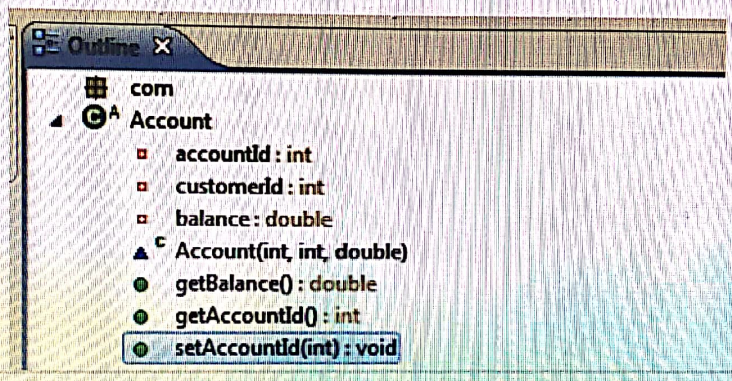
A banking system is looking for automated solution for managing accounts and related deposit and withdrawal. The system would currently support feature of Savings Account, however, it may have more

account types in future. Develop a Java project which serves this system following below guidelines. Class outline is shared after the guidelines. Ensure the same is met 100%, else your solution would be considered null and void.

Create project BankingSystem in eclipse. Create package com within src folder. All classes should be created inside this package.

Create abstract class – Account with attributes account id, customer id and abstract methods – deposit and withdraw. Refer below class outline for this class:

Class outline for Account is given below



Close

Scanned by CamScanner

getBalance() : double
getAccountId() : int
setAccountId(int) : void
getCustomerId() : int
setCustomerId(int) : void
setBalance(double) : void
deposit(double) : double
withdraw(double) : double

Create child class – SavingAccount from this abstract class. This class will implement withdraw and deposit methods where amount to be deposited should be added to the balance and amount to be withdrawn should be deducted from the balance. Withdrawal will not check for zero balance, hence balance can go in negative range. Refer below outline for this class.

Outline

- com
- SavingAccount
  - SavingAccount(int, int, double)
  - deposit(double) : double
  - withdraw(double) : double

Close

Scanned by CamScanner

Create class Bank which has ArrayList of Account. Initialize this list in the constructor.
Write below methods in Bank class:

addAccount – This method will take Account object as a parameter. It will add this object to the list if account with specific id does not exist and return true. If account with specific id already exists, it will return false.

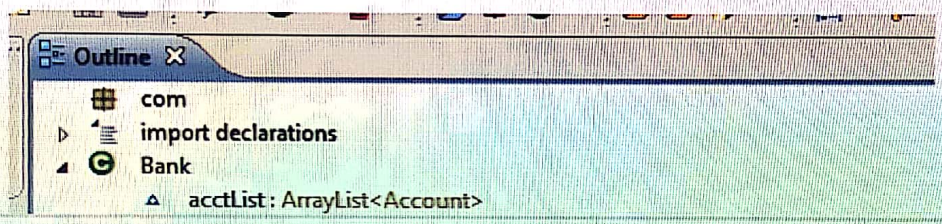
deposit – This method will take account id and deposit amount. Amount will be deposited into specific account with that id. Method will return new balance if account with that id exists, else returns -1.

withdraw – This method will take account id and withdraw amount. It will return -1 if account with specific id does not exist. Else, it will check if withdraw amount is less than or equal to existing balance.

If yes, perform withdrawal .. However, if withdraw amount is greater than existing balance, then exception of class "OutOfBalanceException" should be thrown.If the id is exits the system has to return the account balance

getUniqueCustomerIds() – This method will return a Set of Integer types, where it will contain unique customer ids across all account objects.

acctMap() – This method will return HashMap with account id as key and respective account object as value.



Scanned by CamScanner

Declare class OutOfBlanceException which extends E
Constructor of this class will take existing balance as
parameter. getMessage() method should be overridde
withdraw amount as a String. Outline of this class is a

Scanned by CamScanner

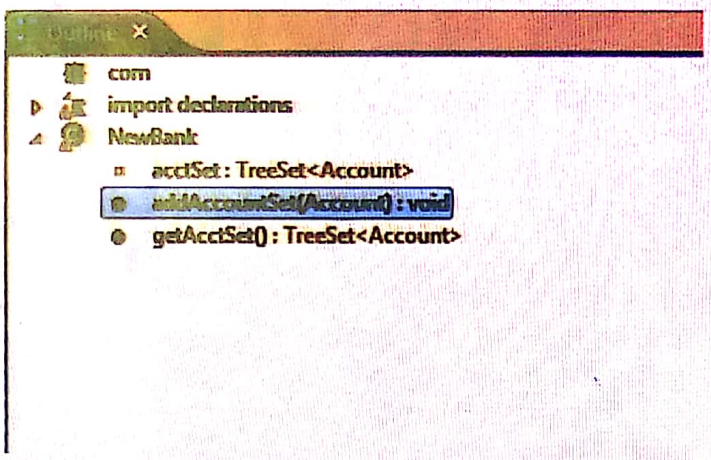
com
OutOfBlanceException
currentAmount : double
withdrawAmount : double
OutOfBlanceException(double, double)
getMessage() : String

Declare class NewBank which has tree set of Account objects. Write method to add account object to this set and returns size of the set after adding the object. Tree set of accounts should maintain accounts in ascending order of account id.(Hint you would need to implement comparable in Account class)

Refer class outline as below:

Close

Scanned by CamScanner



Main class can be constructed for testing all the methods.

Scanned by CamScanner

Create a class Faculty (along with the attributes and methods) as shown in the below outline:

```
com
  Faculty
    empid : int
    name : String
    subject : String
    Faculty(int, String, String)
    getEmpid() : int
    setEmpid(int) : void
    getName() : String
    setName(String) : void
    getSubject() : String
    setSubject(String) : void
```

Note: The constructor Faculty takes empid, name and subject respectively as input parameters.

Create a class Student (along with the attributes and methods) as shown in the below outline:

```
com
  Student
    rollNum : int
    name : String
    marks : float
    Student(int, String, float)
```

Scanned by CamScanner

setName(String) : void
getMarks() : float
setMarks(Float) : void
hashCode() : int
equals(Object) : boolean

Create a class SearchMap as shown in the below outline. The methods are described in the section following the outline.

```
com
  import declarations
  SearchMap
    studentFacultyMap : HashMap<Student, Faculty>
    studentFacultyNameMap : HashMap<String, String>
    getFaculty(Student) : Faculty
    getFacultyName(String) : String
    mapNames(String, String) : void
    mapFaculty(Student, Faculty) : void
```

Close

Scanned by CamScanner

`studentFacultyMap`: HashMap holding the Student and Faculty objects as key value pairs

`studentFacultyNameMap`: HashMap holding the names of the student and name of the faculty as key value pairs.

`mapNames`: This method takes the name of the Student and name of the faculty respectively as input parameters and adds them to the HashMap *studentFacultyNameMap* (name of the student is key and name of the faculty is the value).

`mapFaculty`: This method takes an object of Student and object of Faculty as input and adds them to the HashMap *studentFacultyMap* (where Student object is key and Faculty object is value).

`getFacultyName`: This method takes the name of the student as input and returns the corresponding name of the faculty that is mapped with the Student. If name of the student is not found in the HashMap then the method returns null.

`getFaculty`: This method takes an object of Student as input and returns the corresponding Faculty object that is mapped with this Student. If the student object is not found in the HashMap then the method returns null.

Close

Scanned by CamScanner

list of Employee class.

Write below methods in this class:

1. addEmployee – This method will take Employee reference as parameter and add the same to the employees list after checking if employee with same id does not exist. It will return total employees count if addition is successful, else return -1.
2. searchEmployee – this method will take employee name as parameter and return the total count of employees matching this name.
3. [redacted] - this method will return employee id with maximum total salary. It will return -1 if no employee exists.

Try these methods using sample input/output shared after the class outline:

Outline (In addition to previous question):

```
EmployeeDemo
    main(String[]) : void
Organization
    employees : ArrayList<Employee>
    Organization()
    addEmployee(Employee) : int
    searchEmployee(String) : int
    getEmployeeIdWithMaxSalary() : int
```

Sample input/output:

Close

Scanned by CamScanner

Sample input/output:

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

    Organization o = new Organization();

    System.out.println(o.addEmployee(new Associate(1, "aaa", 5000, 7000)));
    System.out.println(o.addEmployee(new Associate(2, "aaa", 4000, 6000)));
    System.out.println(o.addEmployee(new Consultant(3, "bbb1", 4000, 1000, 5000)));
    System.out.println(o.addEmployee(new Consultant(1, "bbb2", 3000, 1000, 5000)));
    System.out.println(o.addEmployee(new Consultant(4, "bbb3", 3000, 8000, 2000)));

    System.out.println(o.searchEmployee("aaa"));
    System.out.println(o.searchEmployee("bbb"));
    System.out.println(o.getEmployeeIdWithMaxSalary());

}
```

Console

```
<terminated> EmployeeDemo [Java Application] C:\Program Files\Java\jdk1.7.0_60\bin\javaw.exe (Mar 2, 2015 6:38:17 PM)
1
2
3
-1
4
2
0
4
```

Close

Scanned by CamScanner

Create table test_table_[illegible] in Oracle database using dashboard credentials. This table will have two columns - test_id NUMBER(5) PRIMARY KEY, test_name VARCHAR(100)

Insert few records in this table (Manually. Not through JDBC).

Create Java classes as per below templates and instructions in package com:

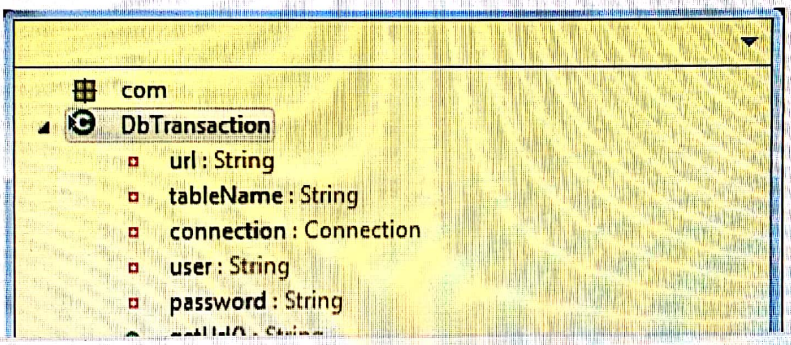
DBTransaction class:

This class has attributes – url, tableName, connection, user and password.

The constructor will take parameters as below sequence:

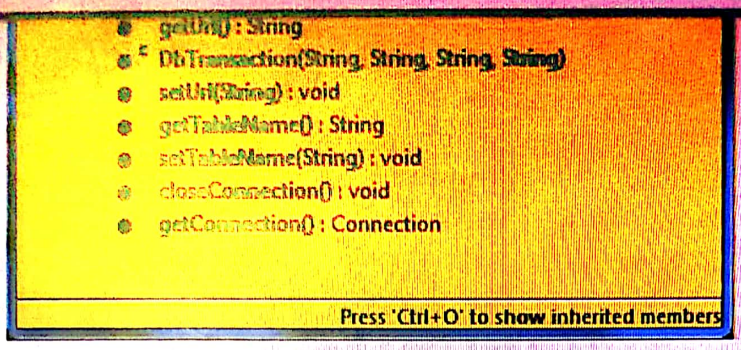
url, user, password, tableName

getConnection and closeConnection should be implemented as per code shared at the end of the question.



Scanned by CamScanner

Question



Test class: this class will have two attributes – id and title. It will work as bean class.

TestProcedure class:

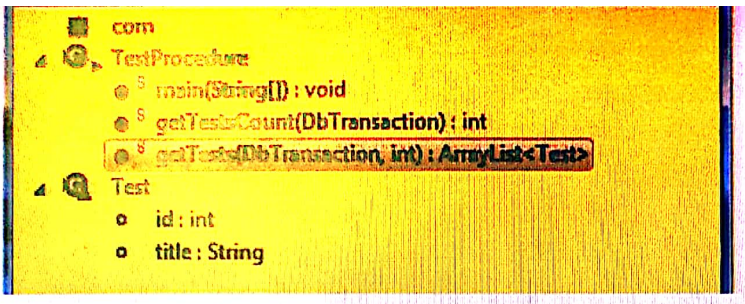
This class has three methods:

getTestsCount: This method will take DbTransaction object and return the total no of records in specified table in that object.

getTests: This method will take DbTransaction object and id value. It will return all records where test_id is greater than specified id. Records are returned as ArrayList of Test class.

Close

Scanned by CamScanner



Submit all Java files in iASCERT tool for assessment.

Code for getConnection() method:

```
public Connection getConnection()
{
        try {
                closeConnection();
                Class.forName("oracle.jdbc.driver.OracleDriver");

                connection = DriverManager.getConnection(url,user,password);

        } catch (SQLException e) {
                // TODO Auto-generated catch block
```

Close

Scanned by CamScanner

```
                connection = [illegible]

        } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();

        } catch (ClassNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();

        }
        return connection;

}
```

Code for closeConnection() method:

```
public  void closeConnection()
{
        try
        {
                if(connection != null  && connection.isClosed() == false)
                        connection.close();

                connection = null;
        }
        catch(SQLException e)
        {
                e.printStackTrace();
        }
}
```

Scanned by CamScanner