**Java Assignment**

**Lab 2: Language Fundamentals, Classes and Objects**

2.1 Write a java program to print person details in the format as shown below:

Person Details:

\_\_\_\_\_\_\_\_\_\_\_\_

First Name: Divya

Last Name: Bharathi

Gender: F

Age: 20

Weight: 85.55

A) **package** com.capg.assignments;

**public** **class** a1{

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

System.***out***.println("Person Details:");

System.***out***.println("\_\_\_\_\_\_\_\_\_\_\_\_\_");

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

System.***out***.println("First Name: Divya");

System.***out***.println("Last Name: Bharathi");

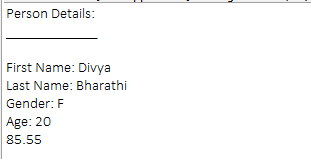
System.***out***.println("Gender: F");

System.***out***.println("Age: 20");

System.***out***.println("85.55");

}

}



2.2: Write a program to accept a number from user as a command line argument and check whether the given number is positive or negative number.

A) **package** com.capg.assignments;

**public** **class** a2{

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

**int** x=Integer.*parseInt*(args[0]);

**if**(x>0){

System.***out***.println("The number is positive");

}**else** **if**(x<0){

System.***out***.println("The number is negative");

}**else**{

System.***out***.println("Enter a valid number");

}

}

}



2.3: Refer the class diagram given below and create a person class.



Create default and parameterized constructor for Person class.

Also Create “PersonMain.java” program and write code for following operations:

1. Create an object of Person class and specify person details through constructor.
2. Display the details in the format given in Lab assignment 2.1

A) Person.java:

**package** com.capgemini.ass.java;

**public** **class** Person {

**private** String firstName;

**private** String lastName;

**private** **char** gender;

//The constructor method

**public** Person(String firstName, String lastName, **char** gender) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.gender = gender;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** **char** getGender() {

**return** gender;

}

**public** **void** setGender(**char** gender) {

**this**.gender = gender;

}

//Method to display the details of the person

**public** **void** displayPersonDetails(){

System.***out***.println("Name: "+firstName);

System.***out***.println("Surname: "+lastName);

System.***out***.println("Gender: "+gender);

}

}

PersonMain.java:

**package** com.capgemini.ass.java;

**public** **class** PersonMain {

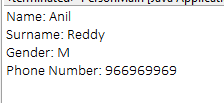
**public** **static** **void** main(String[] args) {

Person p = **new** Person("Anil","Reddy",'M');

p.displayPersonDetails();

}

}



2.4: Modify Lab assignment 2.3 to accept phone number of a person. Create a new method to implement the same and also define method for displaying person details.

Person.java:

**package** com.capgemini.ass.java;

**public** **class** Person {

**private** String firstName;

**private** String lastName;

**private** **char** gender;

**private** **int** phoneNumber;

//The constructor method

//Method to display the details of the person

**public** **void** displayPersonDetails(){

System.***out***.println("Name: "+firstName);

System.***out***.println("Surname: "+lastName);

System.***out***.println("Gender: "+gender);

System.***out***.println("Phone Number: "+phoneNumber);

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** **char** getGender() {

**return** gender;

}

**public** **void** setGender(**char** gender) {

**this**.gender = gender;

}

**public** **int** getPhoneNumber() {

**return** phoneNumber;

}

**public** **void** setPhoneNumber(**int** phoneNumber) {

**this**.phoneNumber = phoneNumber;

}

**public** Person(String firstName, String lastName, **char** gender,

**int** phoneNumber) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.gender = gender;

**this**.phoneNumber = phoneNumber;

}

}

PersonMain.java:

**package** com.capgemini.ass.java;

**public** **class** PersonMain {

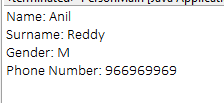
**public** **static** **void** main(String[] args) {

Person p = **new** Person("Anil","Reddy",'M',966969969);

p.displayPersonDetails();

}

}



2.5: Modify the above program, to accept only ‘M’ or ‘F’ as gender field values. Use Enumeration for implementing the same.

Person.java:

**package** com.capgemini.ass.java;

**public** **class** Person {

**private** String firstName;

**private** String lastName;

**private** Gender gender;

**private** **int** phoneNumber;

//The constructor method

//Method to display the details of the person

**public** **void** displayPersonDetails(){

System.***out***.println("Name: "+firstName);

System.***out***.println("Surname: "+lastName);

System.***out***.println("Gender: "+gender);

System.***out***.println("Phone Number: "+phoneNumber);

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** Gender getGender() {

**return** gender;

}

**public** **void** setGender(Gender gender) {

**this**.gender = gender;

}

**public** **int** getPhoneNumber() {

**return** phoneNumber;

}

**public** **void** setPhoneNumber(**int** phoneNumber) {

**this**.phoneNumber = phoneNumber;

}

**public** Person(String firstName, String lastName, Gender gender,

**int** phoneNumber) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.gender = gender;

**this**.phoneNumber = phoneNumber;

}

}

PersonMain.java:

**package** com.capgemini.ass.java;

**public** **class** PersonMain {

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

Person p = **new** Person("Anil","Reddy",Gender.***M***,966969969);

p.displayPersonDetails();

}

}

Gender.java:

**package** com.capgemini.ass.java;

**public** **enum** Gender {

***M***,***F***

}

**Lab 3: Exploring Basic Java Class Libraries**

3.1: Create a method which can perform a particular String operation based on the user’s choice. The method should accept the String object and the user’s choice and return the output of the operation.

Options are

* Add the String to itself
* Replace odd positions with #
* Remove duplicate characters in the String
* Change odd characters to upper case

**package** com.capgemini.ass.java;

**import** java.util.Scanner;

**public** **class** ThreePointOne {

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

@SuppressWarnings("resource")

Scanner input = **new** Scanner(System.***in***);

System.***out***.println("Type your name below:");

String name = input.next();

System.***out***.println(name+" "+name);

}

}

**package** com.capgemini.ass.java;

**public** **class** ThreePointOnePointOne {

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

String a = "capgeminiassignment";

**for** (**int** i=0; i < a.length(); i++){

**if** (i % 2 != 0){

a = a.substring(0,i-1) + "#" + a.substring(i, a.length());

}

}

System.***out***.println(a);

}

}

**package** com.capgemini.ass.java;

**import** java.util.LinkedHashSet;

**import** java.util.Set;

**public** **class** ThreePointOnePointTwo {

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

String string = "camilacabello";

**char**[] chars = string.toCharArray();

Set<Character> charSet = **new** LinkedHashSet<Character>();

**for** (**char** c : chars) {

charSet.add(c);

}

StringBuilder sb = **new** StringBuilder();

**for** (Character character : charSet) {

sb.append(character);

}

System.***out***.println(sb.toString());

}

}

**package** com.capgemini.ass.java;

**public** **class** StringFunctionTester {

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

StringFunctions str=**new** StringFunctions();

String a="hello ";

System.***out***.println(str.add(a));

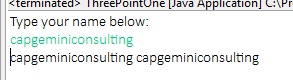
System.***out***.println(str.removeDuplicate(a));

System.***out***.println(str.replace(a));

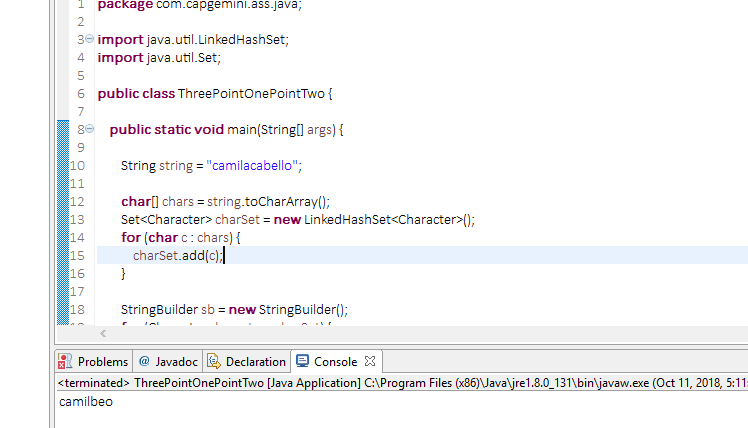
System.***out***.println(str.oddUpper(a));

}

}







3.2: Create a method that accepts a String and checks if it is a positive string. A string is considered a positive string, if on moving from left to right each character in the String comes after the previous characters in the Alphabetical order. For Example: ANT is a positive String (Since T comes after N and N comes after A). The method should return true if the entered string is positive.

A)

**StringChecker.java**

**package** com.capgemini.ass.java;

**public** **class** StringChecker {

**public** String isPositive(String string) {

**int** count=0;

**char**[] chars = string.toCharArray();

**for** (**int** i = 0; i < chars.length-1; i++) {

**if**(chars[i]<chars[i+1]){

count++;

}

}

**if**(count==chars.length-1){

**return** "Positive";

}

**return** "Negative";

}

}

**Stringtester.java**

**package** com.capgemini.ass.java;

**public** **class** StringTester {

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

StringChecker str=**new** StringChecker();

System.***out***.println(str.isPositive("ANT"));

}

}



3.3: Create a method to accept date and print the duration in days, months and years with regards to current system date.

A)

**package** com.capgemini.ass.java;

**import** java.time.LocalDate;

**import** java.time.Month;

**import** java.time.Period;

**public** **class** DateDemo1 {

**public** **void** date(LocalDate d) {

LocalDate today=LocalDate.*now*();

Period period=d.until(today);

System.***out***.println("Months:"+period.getMonths());

System.***out***.println("Days:"+period.getDays());

System.***out***.println("Years:"+period.getYears());

}

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

DateDemo1 datedemo=**new** DateDemo1();

LocalDate d=LocalDate.*of*(2014, Month.***SEPTEMBER***, 8);

datedemo.date(d);

}

}



3.4: Revise exercise 3.3 to accept two LocalDates and print the duration between dates in days, months and years.

A)

package com.capgemini.ass.java;

import java.time.LocalDate;

import java.time.Month;

import java.time.Period;

public class DateDemo {

public void date(LocalDate d1,LocalDate d2) {

Period period=d1.until(d2);

System.out.println("Months:"+period.getMonths());

System.out.println("Days:"+period.getDays());

System.out.println("Years:"+period.getYears());

}

public static void main(String[] args) {

DateDemo datedemo=new DateDemo();

LocalDate d1=LocalDate.of(2014, Month.SEPTEMBER,8);

LocalDate d2=LocalDate.of(2018, Month.MARCH, 10);

datedemo.date(d1,d2);

}

}



3.5: Create a method to accept product purchase date and warrantee period (in terms of months and years). Print the date on which warrantee of product expires.

A)

package com.capgemini.ass.java;

import java.util.Calendar;

import java.util.GregorianCalendar;

public class WarrantyChecker {

public void check(Calendar c1,int months) {

Calendar c2=new GregorianCalendar(c1.get(Calendar.YEAR),c1.get(Calendar.DATE),c1.get(Calendar.MONTH)+months);

System.out.println(c2.get(Calendar.YEAR)+","+c2.get(Calendar.MONTH));

}

public static void main(String[] args) {

WarrantyChecker wc=new WarrantyChecker();

Calendar c1=new GregorianCalendar(2018,28,Calendar.MAY);

int months=20;

wc.check(c1, months);

}

}



3.6: Create a method which accept zone id and print the current date and time with respect to given zone. (Hint: Few zones to test your code. America/New\_York, Europe/London, Asia/Tokyo, US/Pacific, Africa/Cairo, Australia/Sydney etc.)

A)

package com.capgemini.ass.java;

import java.time.ZoneId;

import java.time.ZonedDateTime;

public class ZoneTester {

public void zone(String id) {

ZonedDateTime t=ZonedDateTime.now(ZoneId.of(id));

System.out.println(t);

}

public static void main(String[] args) {

ZoneTester test=new ZoneTester();

test.zone("America/New\_York");

}

}



3.7: Modify Lab assignment 2.3 to perform following functionalities:

1. Add a method called calculateAge which should accept person’s date of birth and calculate age of a person.
2. Add a method called getFullName(String firstName, String lastName) which should return full name of a person

Display person details with age and fullname.

A)

**Persons.java**

package com.capgemini.ass.java;

import java.time.LocalDate;

import java.time.Period;

public class Persons {

private String firstName;

private String lastName;

private Gender gender;

private int phoneNumber;

private LocalDate dob;

public Persons(){

}

public Persons(String firstName, String lastName, Gender gender,

int phoneNumber, LocalDate dob) {

super();

this.firstName = firstName;

this.lastName = lastName;

this.gender = gender;

this.phoneNumber = phoneNumber;

this.dob = dob;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public Gender getGender() {

return gender;

}

public void setGender(Gender gender) {

this.gender = gender;

}

public int getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(int phoneNumber) {

this.phoneNumber = phoneNumber;

}

public LocalDate getDob() {

return dob;

}

public void setDob(LocalDate dob) {

this.dob = dob;

}

public String display() {

return "FirstName=" + firstName + ", LastName=" + lastName

+ ", Gender=" + gender + ", PhoneNumber=" + phoneNumber ;

}

public int age(LocalDate dob) {

LocalDate today=LocalDate.now();

Period period=dob.until(today);

return period.getYears();

}

public String getFullName(String firstName,String lastName) {

return firstName+" "+lastName;

}

}

**PersonMain.java**

package com.capgemini.ass.java;

import java.time.LocalDate;

import java.time.Month;

public class PersonMain {

public static void main(String[] args) {

LocalDate dob=LocalDate.of(1997, Month.DECEMBER, 11);

Persons person=new Persons("Raj\_Kamal","Chenumalla",Gender.M,987654321,dob);

System.out.println(person.display());

System.out.println("Age:"+person.age(dob));

System.out.println("Full Name:"+person.getFullName("Raj\_Kamal","Chenumalla"));

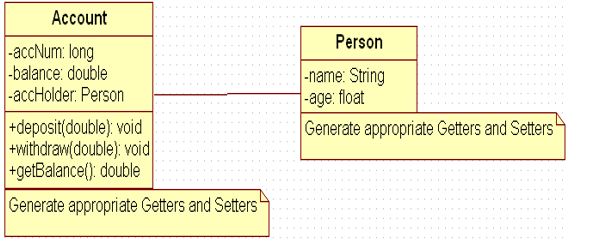
}

}



**Inheritance and Polymorphism**

4.1: Refer the case study 1in Page No: 5 and create Account Class as shown below in class diagram. Ensure minimum balance of INR 500 in a bank account is available.



1. Create Account for smith with initial balance as INR 2000 and for Kathy with initial balance as 3000. (accNum should be auto generated).
2. Deposit 2000 INR to smith account.
3. Withdraw 2000 INR from Kathy account.
4. Display updated balances in both the account.

Generate toString() method.

A) Account.java

//Account.java

**package** com.capgemini.trg.service;

**public** **class** Account **extends** PersonInheritance {

**private** Long accNum;

**private** Double balance;

**private** PersonInheritance accHolder;

**public** Account() {

**super**();

}

**public** Account(Long accNum, Double balance, PersonInheritance accHolder) {

**super**();

**this**.accNum = accNum;

**this**.balance = balance;

**this**.accHolder = accHolder;

}

**public** Long accNumAutoGeneration(){

**this**.accNum+=1000L;

**return** **this**.accNum;

}

**public** Long getAccNum() {

**return** accNum;

}

**public** **void** setAccNum(Long accNum) {

**this**.accNum = accNum;

}

**public** Double getBalance() {

**return** balance;

}

**public** **void** setBalance(Double balance) {

**this**.balance = balance;

}

**public** PersonInheritance getAccHolder() {

**return** accHolder;

}

**public** **void** setAccHolder(PersonInheritance accHolder) {

**this**.accHolder = accHolder;

}

**public** **void** deposit(Double amount){

**this**.balance=**this**.balance+amount;

}

**public** **double** withdraw(Double amount){

**if**(**this**.getBalance()<=0.0){

System.***out***.println("No balance in your Current Account");

**return** -1.0;

}**else** **if**(**this**.getBalance()<amount){

System.***out***.println("Insufficient funds in your Current Account");

**return** 0.0;

}**else** **if**((**this**.getBalance()-amount)<500.00){

System.***out***.println("Minimum balance of Rs.500 should be maintained in your current Account");

}

**this**.setBalance(**this**.getBalance()-amount);

**return** amount;

}

@Override

**public** String toString() {

**return** "Account [accNum=" + accNum + ", balance=" + balance

+ ", accHolder=" + accHolder + "]";

}

}

PersonInheritance.java

//PersonInheritance.java

**package** com.capgemini.trg.service;

**public** **class** PersonInheritance {

**private** **float** age;

**private** String name;

**public** PersonInheritance() {

**super**();

}

**public** PersonInheritance(**float** age, String name) {

**super**();

**this**.age = age;

**this**.name = name;

}

**public** **float** getAge() {

**return** age;

}

**public** **void** setAge(**float** age) {

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String toString() {

**return** "PersonInheritance [age=" + age + ", name=" + name + "]";

}

}

AccountTester.java

//AccountTester.java

**package** com.capgemini.trg.ui;

**import** com.capgemini.trg.service.Account;

**import** com.capgemini.trg.service.PersonInheritance;

**public** **class** AccountTester {

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

PersonInheritance person=**new** PersonInheritance(26,"Smith");

PersonInheritance person1=**new** PersonInheritance(26,"Kathy");

Account account = **new** Account((**long**)(Math.*random*()\*100000),3000.0,person);

Account account1=**new** Account((**long**)(Math.*random*()\*100000),3000.0,person1);

System.***out***.println("Account Details of smith: "+account);

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

System.***out***.println("Account Details of kathy: "+account1);

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

account.deposit(2000.00);

System.***out***.println("after Rs.2000.0 deposit into smith account");

System.***out***.println(account);

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

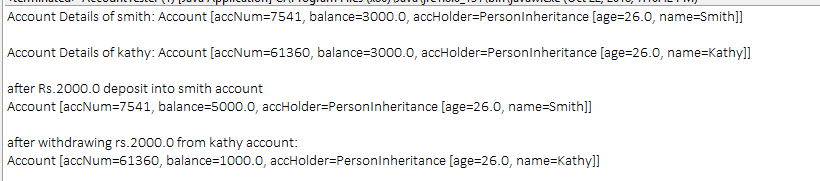
System.***out***.println("after withdrawing rs.2000.0 from kathy account:");

account1.withdraw(2000.0);

System.***out***.println(account1);

}

}



4.2: Extend the functionality through Inheritance and polymorphism (Maintenance)

Inherit two classes Savings Account and Current Account from account class. Implement the following in the respective classes.

1. Savings Account
2. Add a variable called minimum Balance and assign final modifier.
3. Override method called withdraw (This method should check for minimum balance and allow withdraw to happen)
4. Current Account
5. Add a variable called overdraft Limit
6. Overridemethod called withdraw (checks whether overdraft limit is reached and returns a boolean value accordingly)

Account.java

**package** com.capgemini.trg.service;

**public** **class** Account **extends** PersonInheritance {

Long accNum;

Double balance;

PersonInheritance accHolder;

Long count =0L;

**public** Account() {

**super**();

accNum = count++;

}

**public** Account(Long accNum, Double balance, PersonInheritance accHolder) {

**super**();

**this**.accNum = accNum;

**this**.balance = balance;

**this**.accHolder = accHolder;

}

**public** Long accNumAutoGeneration(){

**this**.accNum+=1000L;

**return** **this**.accNum;

}

**public** Long getAccNum() {

**return** accNum;

}

**public** **void** setAccNum(Long accNum) {

**this**.accNum = accNum;

}

**public** Double getBalance() {

**return** balance;

}

**public** **void** setBalance(Double balance) {

**this**.balance = balance;

}

**public** PersonInheritance getAccHolder() {

**return** accHolder;

}

**public** **void** setAccHolder(PersonInheritance accHolder) {

**this**.accHolder = accHolder;

}

**public** **void** deposit(Double amount){

**this**.balance=**this**.balance+amount;

}

**public** **double** withdraw(Double amount){

**if**(**this**.balance<=0.0){

**return** -1.0;

}**else** **if**(**this**.balance<amount){

**return** 0.0;

}**else**{

**this**.balance=**this**.balance-amount;

**return** amount;

}

}

@Override

**public** String toString() {

**return** "Account [accNum=" + accNum + "\n"+

" balance=" + balance+"\n"+

", accHolder=" + accHolder + "]";

}

**public** Long accNumAutoGeneration(Long accNum){

Long counter=1000L;

**this**.accNum=**this**.accNum+counter;

counter++;

**return** accNum;

}

}

SavingsAccount.java

**package** com.capgemini.trg.service;

**import** java.time.LocalDate;

**public** **class** SavingsAccount **extends** Account {

**public** **static** **final** **double** ***minimumBalance***=500.00;

**public** SavingsAccount() {

**super**();

}

**public** SavingsAccount(Long accNum, Double balance,

PersonInheritance accHolder) {

**super**(accNum, balance, accHolder);

}

@Override

**public** String toString() {

**return** "SavingsAccount [accNum=" + accNum + ", balance=" + balance

+ ", AccHolder=" +accHolder + ", getAccNum()=" + getAccNum()

+ ", getBalance()=" + getBalance() + ", getAccHolder()="

+ getAccHolder() + ", toString()=" + **super**.toString()

+ ", getClass()=" + getClass() + ", hashCode()=" + hashCode()

+ "]";

}

@Override

**public** **double** withdraw(Double amount) {

**if**(**this**.getBalance()<=0.0){

System.***out***.println("No balance in your Current Account");

**return** -1.0;

}**else** **if**(**this**.getBalance()<amount){

System.***out***.println("Insufficient funds in your Current Account");

**return** 0.0;

}**else** **if**((**this**.getBalance()-amount)<500.00){

System.***out***.println("Minimum balance of Rs.500 should be maintained in your current Account");

}

**this**.setBalance(**this**.getBalance()-amount);

**return** amount;

}

}

CurrentAccount.java

**package** com.capgemini.trg.service;

**public** **class** CurrentAccount **extends** Account {

**public** **static** **final** **double** ***overDraftLimit***=3000.00;

**public** CurrentAccount() {

**super**();

}

**public** CurrentAccount(Long accNum, Double balance,

PersonInheritance accHolder) {

**super**(accNum, balance, accHolder);

// **TODO** Auto-generated constructor stub

}

@Override

**public** String toString() {

**return** "Savings Account Details..\n"+**super**.toString()+

"[OverDraftLimit = "+***overDraftLimit***+"]";

}

@Override

**public** **double** withdraw(Double amount) {

**if**(**this**.getBalance()<=0.0){

System.***out***.println("No balance in your Current Account");

**return** -1.0;

}**else** **if**(**this**.getBalance()<amount){

System.***out***.println("Insufficient funds in your Current Account");

**this**.setBalance(**this**.getBalance()-amount);

**return** 0.0;

}**else** **if**(amount>=***overDraftLimit***){

**boolean** bl =LimitCrossed();

System.***out***.println(bl);

}**else** **if**((**this**.getBalance()-amount)<500.00){

System.***out***.println("Minimum balance of Rs.500 should be maintained in your current Account");

**this**.setBalance(**this**.getBalance()-amount);

}

**return** amount;

}

**private** **boolean** LimitCrossed() {

System.***out***.println("Sorry!!...Your overdraft limit of 3000.00 is reached");

**return** **false**;

}

}

AccountTester.java

package com.capgemini.trg.ui;

import java.util.Scanner;

import com.capgemini.trg.service.Account;

import com.capgemini.trg.service.CurrentAccount;

import com.capgemini.trg.service.PersonInheritance;

import com.capgemini.trg.service.SavingsAccount;

public class AccountTester {

public static void main(String args[]){

PersonInheritance person=**new** PersonInheritance(26,"Smith");

Account account=**new** SavingsAccount();

Account account1=**new** CurrentAccount();

account = **new** SavingsAccount(account.accNumAutoGeneration(),5000.00,person);

account1 = **new** CurrentAccount(account1.accNumAutoGeneration(),5000.00,person);

System.***out***.println("Account Details of Smith: "+account);

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

Scanner sc = **new** Scanner(System.***in***);

try{

System.***out***.println("Choose account:1.Savings 2.Current:");

**int** ch =sc.nextInt();

System.***out***.println("Enter how much money you want to withdraw:");

**double** amount = sc.nextInt();

**if**(ch==1){

System.***out***.println("withdrwal amount =Rs."+amount);

System.***out***.println("Balance before withdrwal in your Savings Account=Rs" +account.getBalance());

account.withdraw((**double**) amount);

System.***out***.println("Balance after withdrwal in your Savings Account=Rs" +account.getBalance());

}

**else** **if**(ch==2){

account1.withdraw((**double**) amount);

System.***out***.println("withdrwal amount =Rs."+amount);

System.***out***.println("Balance before Transaction in your Current Account=Rs" +account1.getBalance());

System.***out***.println("Balance after Transaction in your Current Account=Rs" +account1.getBalance());

}

**else**{

System.***out***.println("Your option of selection is wrong Choose correct one");

}

}catch(Exception e){

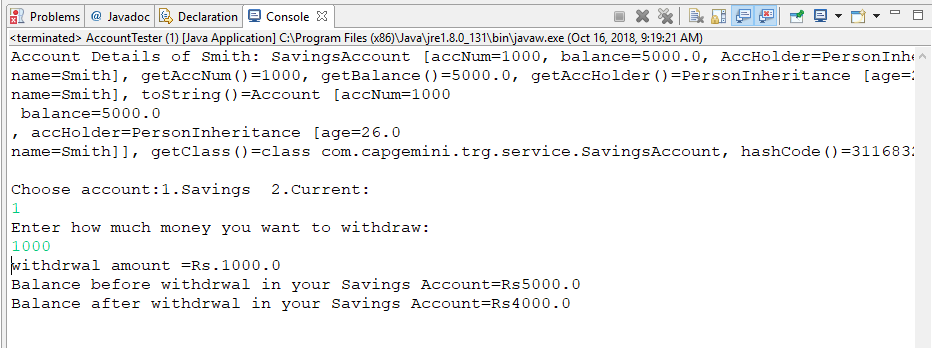
System.out.println(“Exception occured”+e);

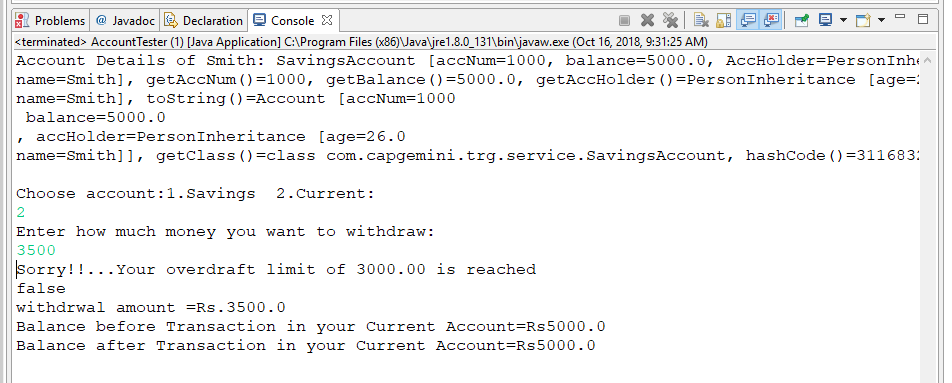
}

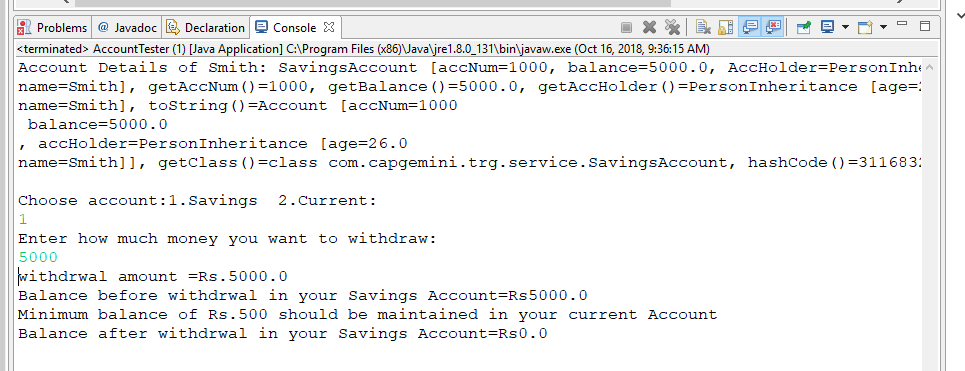
sc.close();

}

}







Abstract classes and Interfaces

5.1: Refer the case study 2 in page no: 5 and create an application for that requirement by creating packages and classes as given below:

**com.cg.eis.bean**

In this package, create “Employee” class with different attributes such as id, name, salary, designation, insuranceScheme.

**com.cg.eis.service**

This package will contain code for services offered in Employee Insurance System. The service class will have one EmployeeService Interface and its corresponding implementation class.

**com.cg.eis.pl**

This package will contain code for getting input from user, produce expected output to the user and invoke services offered by the system.

The services offered by this application currently are:

1. Get employee details from user.
2. Find the insurance scheme for an employee based on salary and designation.
3. Display all the details of an employee.

Employee.java

**package** com.cg.eis.bean;

**public** **class** Employee {

**private** Long id;

**private** String name;

**private** Double salary;

**private** String designation;

**private** String insuranceScheme;

**public** Employee() {

**super**();

}

**public** Employee(Long id, String name, Double salary, String designation) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

**this**.designation = designation;

}

**public** Employee(Long id, String name, Double salary, String designation,

String insuranceScheme) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

**this**.designation = designation;

**this**.insuranceScheme = insuranceScheme;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Double getSalary() {

**return** salary;

}

**public** **void** setSalary(Double salary) {

**this**.salary = salary;

}

**public** String getDesignation() {

**return** designation;

}

**public** **void** setDesignation(String designation) {

**this**.designation = designation;

}

**public** String getInsuranceScheme() {

**return** insuranceScheme;

}

**public** **void** setInsuranceScheme(String insuranceScheme) {

**this**.insuranceScheme = insuranceScheme;

}

@Override

**public** String toString() {

**return** "Employee details.."+"\n"+

"id=" + getId() +"\n"+

"name=" + getName() +"\n"+

"salary=" + getSalary()+"\n"+

"designation=" + getDesignation() +"\n"+

"insuranceScheme="+ getInsuranceScheme();

}

}

EmployeeInsuranceScheme.java

**package** com.cg.eis.service;

**import** com.cg.eis.bean.Employee;

**public** **interface** EmployeeInsuranceInterface {

**public** **abstract** String employeeInsuranceSchemeOffers(Employee e);

}

User.java

**package** com.cg.eis.pl;

**import** java.util.Scanner;

**import** com.cg.eis.bean.Employee;

**import** com.cg.eis.service.EmployeeInsuranceInterface;

**public** **class** User **implements** EmployeeInsuranceInterface {

Employee employee;

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

Scanner scanner=**new** Scanner(System.***in***);

System.***out***.println("Enter employee name");

String name=scanner.nextLine();

System.***out***.println("Enter employee designation");

String designation=scanner.nextLine();

System.***out***.println("Enter the employee id");

Long id =scanner.nextLong();

System.***out***.println("Enter employee salary");

Double salary=scanner.nextDouble();

User user=**new** User();

Employee employee1=**new** Employee(id,name,salary,designation);

String insuranceScheme=user.employeeInsuranceSchemeOffers(employee1);

Employee employee2=**new** Employee(id,name,salary,designation,insuranceScheme);

System.***out***.println(employee2);

}

@Override

**public** String employeeInsuranceSchemeOffers(Employee e) {

**if**(e.getSalary()>5000 && e.getSalary()<20000 && e.getDesignation().equalsIgnoreCase("System Associate")){

e.setInsuranceScheme("Scheme C");

}

**else** **if**(e.getSalary()>=20000 && e.getSalary()<40000 && e.getDesignation().equalsIgnoreCase("Programmer")){

e.setInsuranceScheme("Scheme B");

}

**else** **if**(e.getSalary()>=40000 && e.getDesignation().equalsIgnoreCase("Manager")){

e.setInsuranceScheme("Scheme A");

}

**else** **if**(e.getSalary()<=5000 && e.getDesignation().equalsIgnoreCase("Clerk")){

e.setInsuranceScheme("No Scheme");

}

**else**

{

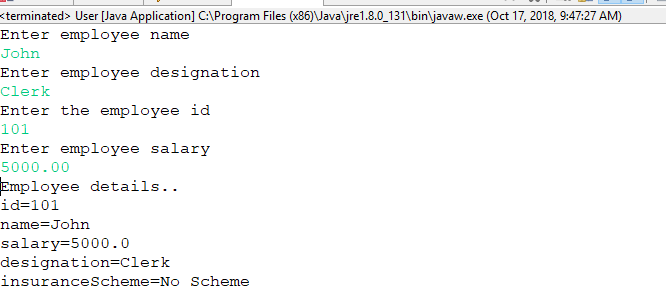
System.***out***.println("Sorry!! Your designation donot have any scheme..");

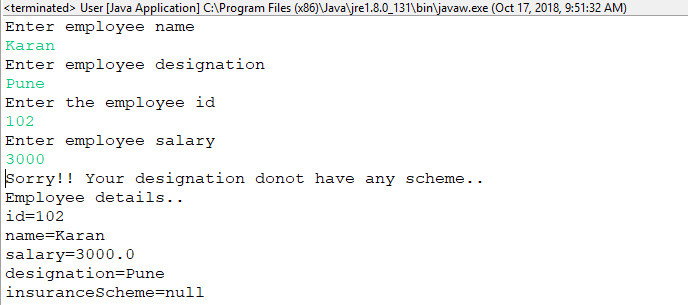
}

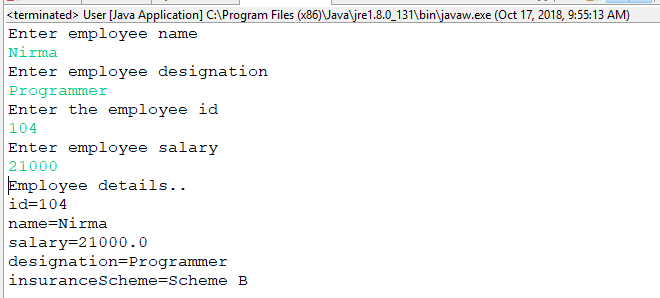
**return** e.getInsuranceScheme();

}

}







5.2: Use overrides annotation for the overridden methods available in a derived class of an interface of all the assignments.

5.3: Refer the problem statement 4.1. Modify account class as abstract class and declare withdraw method.

Account.java

**package** com.capgemini.trg.service;

**public** **abstract** **class** Account **extends** PersonInheritance {

Long accNum;

Double balance;

PersonInheritance accHolder;

Long count =0L;

**public** Account() {

**super**();

accNum = count++;

}

**public** Account(Long accNum, Double balance, PersonInheritance accHolder) {

**super**();

**this**.accNum = accNum;

**this**.balance = balance;

**this**.accHolder = accHolder;

}

**public** Long accNumAutoGeneration(){

**this**.accNum+=1000L;

**return** **this**.accNum;

}

**public** Long getAccNum() {

**return** accNum;

}

**public** **void** setAccNum(Long accNum) {

**this**.accNum = accNum;

}

**public** Double getBalance() {

**return** balance;

}

**public** **void** setBalance(Double balance) {

**this**.balance = balance;

}

**public** PersonInheritance getAccHolder() {

**return** accHolder;

}

**public** **void** setAccHolder(PersonInheritance accHolder) {

**this**.accHolder = accHolder;

}

**public** **void** deposit(Double amount){

**this**.balance=**this**.balance+amount;

}

**public** **abstract** **double** withdraw(Double amount);

@Override

**public** String toString() {

**return** "Account [accNum=" + accNum + "\n"+

" balance=" + balance+"\n"+

", accHolder=" + accHolder + "]";

}

**public** Long accNumAutoGeneration(Long accNum){

Long counter=1000L;

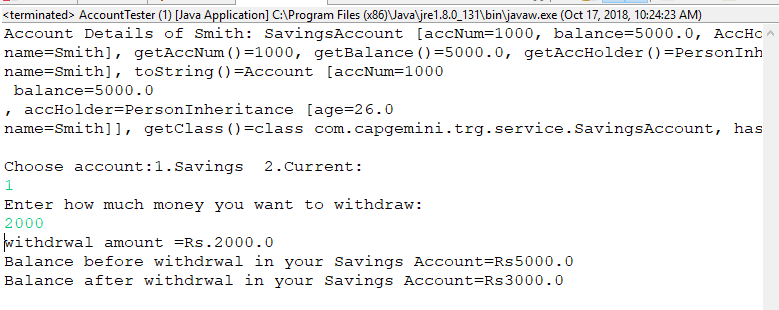
**this**.accNum=**this**.accNum+counter;

counter++;

**return** accNum;

}

}



**Exception Handling**

6.1: Modify the Lab assignment 2.3 to validate the full name of an employee. Create and throw a user defined exception if firstName and lastName is blank.

PersonNameCheckException.java

**package** com.capgemini.trg.service;

**public** **class** PersonNameCheckException **extends** Exception {

**private** String firstName;

**private** String lastName;

**public** PersonNameCheckException() {

**super**();

}

**public** PersonNameCheckException(String firstName, String lastName) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

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

{

System.***out***.println("Person Details:");

System.***out***.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

System.***out***.println("First Nmae: "+getFirstName());

System.***out***.println("Last Name: "+getLastName());

}

@Override

**public** String toString() {

// **TODO** Auto-generated method stub

**return** **super**.toString()+"\n Firstname and Lastname should not be blank";

}

}

PersonNameCheck.java

**package** com.capgemini.trg.service;

**import** java.util.Scanner;

**public** **class** PersonNameCheck {

**private** String firstName;

**private** String lastName;

**public** **void** getpersonDetails() **throws** PersonNameCheckException{

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter first name of the person");

firstName=sc.nextLine();

System.***out***.println("Enter last name of the person");

lastName=sc.nextLine();

String regex="";

**if**(firstName.matches(regex)&&(lastName.matches(regex))){

**throw** **new** PersonNameCheckException();

}

**else**{

**this**.firstName=firstName;

System.***out***.println("FirstName: "+firstName);

**this**.lastName=lastName;

System.***out***.println("Lastname: "+lastName);

}

}

}

PersonNameCheckMain.java

**package** com.capgemini.trg.ui;

**import** com.capgemini.trg.service.PersonNameCheck;

**import** com.capgemini.trg.service.PersonNameCheckException;

**public** **class** PersonNameCheckMain {

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

PersonNameCheck person=**new** PersonNameCheck();

**try** {

person.getpersonDetails();

} **catch** (PersonNameCheckException e) {

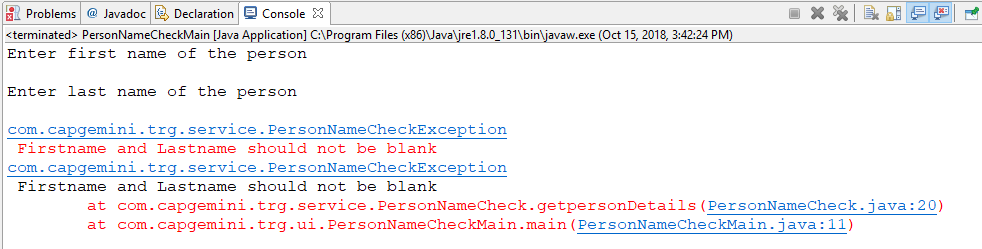
e.printStackTrace();

System.***out***.println(e);

}

}

}



6.2: Validate the age of a person in Lab assignment 4.2 and display proper message by using user defined exception. Age of a person should be above 15.

PersonInheritance.java

**package** com.capgemini.trg.service;

**public** **class** PersonInheritance {

**private** **float** age;

**private** String name;

**public** PersonInheritance() {

**super**();

}

**public** PersonInheritance(**float** age, String name)**throws** AgeVerificationException {

**super**();

**float** temp=age;

**if**(temp<15){

**throw** **new** AgeVerificationException(temp);

}

**this**.age=temp;

**this**.name = name;

}

**public** **void** getAge() {

**this**.age=age;;

}

**public** **void** setAge(**float** age) {

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

@Override

**public** String toString() {

**return** "PersonInheritance [age=" + age + "\n"+

"name=" + name + "]";

}

}

AgeVerificationException.java

**package** com.capgemini.trg.service;

**public** **class** AgeVerificationException **extends** Exception {

**private** **float** age;

**public** AgeVerificationException() {

**super**();

}

**public** AgeVerificationException(**float** age) {

**super**();

**this**.age = age;

}

**public** **float** getAge() {

**return** age;

}

**public** **void** setAge(**float** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** **super**.toString()+"\n Age should be greater than 15 to open an account";

}

}

AccountTester.java

**package** com.capgemini.trg.ui;

**import** java.util.Scanner;

**import** com.capgemini.trg.service.Account;

**import** com.capgemini.trg.service.AgeVerificationException;

**import** com.capgemini.trg.service.CurrentAccount;

**import** com.capgemini.trg.service.PersonInheritance;

**import** com.capgemini.trg.service.SavingsAccount;

**public** **class** AccountTester {

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

Scanner scanner=**new** Scanner(System.***in***);

System.***out***.println("Enter account holder name:");

String name=scanner.nextLine();

System.***out***.println("Enter account holder's age: ");

**float** age=scanner.nextFloat();

PersonInheritance person=**new** PersonInheritance();

**try** {

person = **new** PersonInheritance(age,name);

Account account=**new** SavingsAccount();

Account account1=**new** CurrentAccount();

account = **new** SavingsAccount(account.accNumAutoGeneration(),5000.00,person);

account1 = **new** CurrentAccount(account1.accNumAutoGeneration(),5000.00,person);

System.***out***.println("Account Details of Smith: "+account);

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

Scanner sc = **new** Scanner(System.***in***);

**try**{

System.***out***.println("Choose account:1.Savings 2.Current:");

**int** ch =sc.nextInt();

System.***out***.println("Enter how much money you want to withdraw:");

**double** amount = sc.nextInt();

**if**(ch==1){

System.***out***.println("withdrwal amount =Rs."+amount);

System.***out***.println("Balance before withdrwal in your Savings Account=Rs" +account.getBalance());

account.withdraw((**double**) amount);

System.***out***.println("Balance after withdrwal in your Savings Account=Rs" +account.getBalance());

}

**else** **if**(ch==2){

account1.withdraw((**double**) amount);

System.***out***.println("withdrwal amount =Rs."+amount);

System.***out***.println("Balance before Transaction in your Current Account=Rs" +account1.getBalance());

System.***out***.println("Balance after Transaction in your Current Account=Rs" +account1.getBalance());

}

**else**{

System.***out***.println("Your option of selection is wrong Choose correct one");

}

}**catch**(Exception e ){

System.***out***.println("Exception occured "+e);

}

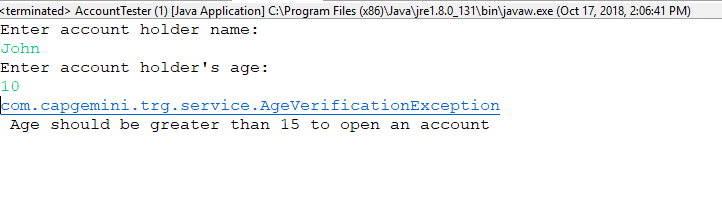
} **catch** (AgeVerificationException e1) {

System.***out***.println(e1);

}

}

}



6.3: Modify the Lab assignment 5.1 to handle exceptions. Create an Exception class named as “EmployeeException”(User defined Exception) in a package named as “com.cg.eis.exception” and throw an exception if salary of an employee is below than 3000. Use Exception Handling mechanism to handle exception properly.

EmployeeException.java

**package** com.cg.eis.exception;

**public** **class** EmployeeException **extends** Exception{

**private** Double salary;

**public** EmployeeException() {

**super**();

}

**public** EmployeeException(Double salary) {

**super**();

**this**.salary = salary;

}

**public** Double getSalary() {

**return** salary;

}

**public** **void** setSalary(Double salary) {

**this**.salary = salary;

}

@Override

**public** String toString() {

**return** "Employee Salary should be greater than Rs.3000.00";

}

}

Employee.java

**package** com.cg.eis.bean;

**import** com.cg.eis.exception.EmployeeException;

**public** **class** Employee {

**private** Long id;

**private** String name;

**private** Double salary;

**private** String designation;

**private** String insuranceScheme;

**public** Employee() {

**super**();

}

**public** Employee(Long id, String name, Double salary, String designation)**throws** EmployeeException {

**super**();

**this**.id = id;

**this**.name = name;

**if**(salary<3000.00){

**throw** **new** EmployeeException();

}

**this**.salary=salary;

**this**.designation = designation;

}

**public** Employee(Long id, String name, Double salary, String designation,

String insuranceScheme) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

**this**.designation = designation;

**this**.insuranceScheme = insuranceScheme;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Double getSalary() {

**return** salary;

}

**public** **void** setSalary(Double salary) {

**this**.salary = salary;

}

**public** String getDesignation() {

**return** designation;

}

**public** **void** setDesignation(String designation) {

**this**.designation = designation;

}

**public** String getInsuranceScheme() {

**return** insuranceScheme;

}

**public** **void** setInsuranceScheme(String insuranceScheme) {

**this**.insuranceScheme = insuranceScheme;

}

@Override

**public** String toString() {

**return** "Employee details.."+"\n"+

"id=" + getId() +"\n"+

"name=" + getName() +"\n"+

"salary=" + getSalary()+"\n"+

"designation=" + getDesignation() +"\n"+

"insuranceScheme="+ getInsuranceScheme();

}

}

User.java

package com.cg.eis.pl;

import java.util.Scanner;

import com.cg.eis.bean.Employee;

import com.cg.eis.exception.EmployeeException;

import com.cg.eis.service.EmployeeInsuranceInterface;

public class User implements EmployeeInsuranceInterface {

Employee employee;

public static void main(String[] args) {

Scanner scanner=new Scanner(System.in);

System.out.println("Enter employee name");

String name=scanner.nextLine();

System.out.println("Enter employee designation");

String designation=scanner.nextLine();

System.out.println("Enter the employee id");

Long id =scanner.nextLong();

System.out.println("Enter employee salary");

Double salary=scanner.nextDouble();

User user=new User();

Employee employee1;

try {

employee1 = new Employee(id,name,salary,designation);

String insuranceScheme=user.employeeInsuranceSchemeOffers(employee1);

Employee employee2=new Employee(id,name,salary,designation,insuranceScheme);

System.out.println(employee2);

} catch (EmployeeException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

@Override

public String employeeInsuranceSchemeOffers(Employee e) {

if(e.getSalary()>5000 && e.getSalary()<20000 && e.getDesignation().equalsIgnoreCase("System Associate")){

e.setInsuranceScheme("Scheme C");

}

else if(e.getSalary()>=20000 && e.getSalary()<40000 && e.getDesignation().equalsIgnoreCase("Programmer")){

e.setInsuranceScheme("Scheme B");

}

else if(e.getSalary()>=40000 && e.getDesignation().equalsIgnoreCase("Manager")){

e.setInsuranceScheme("Scheme A");

}

else if(e.getSalary()<=5000 && e.getDesignation().equalsIgnoreCase("Clerk")){

e.setInsuranceScheme("No Scheme");

}

else

{

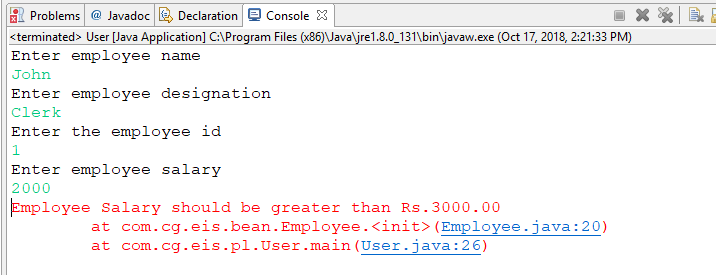
System.out.println("Sorry!! Your designation and corresponding salary donot have any scheme..");

}

return e.getInsuranceScheme();

}

}



**Arrays and Collections**

7.1: Write a program to store product names in a string array and sort strings available in an array.

ProductArray.java

**package** com.capgemini.trg.ui;

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** ProductArray {

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

Scanner scanner=**new** Scanner(System.***in***);

System.***out***.println("Enter number of products ");

**try**{

**int** countOfProducts=scanner.nextInt();

String products[]=**new** String[countOfProducts];

**for**(**int** i=0;i<countOfProducts;i++){

System.***out***.println("Enter name of "+(i+1)+" product");

products[i]=scanner.next();

}

Arrays.*sort*(products);

System.***out***.println("Products in sorted form:");

**for**(String s:products){

System.***out***.println(s);

}

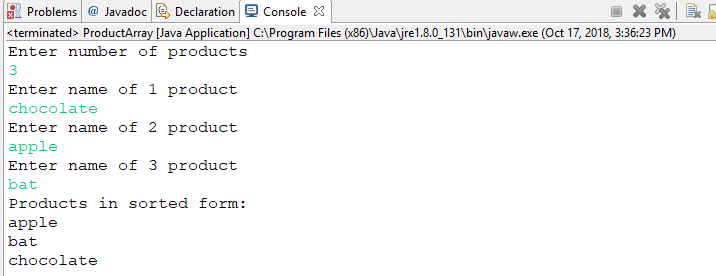
}**catch**(Exception e){

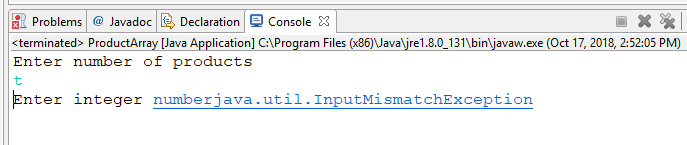
System.***out***.println("Enter integer number"+e);

}

}

}





7.2: Modify the above program to store product names in anArrayList, sort strings available in an arrayList and display the names using for-each loop.

ProductArray.java

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** ProductArray {

**private** **static** Scanner *scanner*=**new** Scanner(System.***in***);

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

List<String> productList=**new** ArrayList<>();

*getProductNames*(productList);

*showProductNames*(productList);

}

**private** **static** **void** showProductNames(List<String> productList) {

String products[]=productList.toArray(**new** String[productList.size()]);

System.***out***.println("Before sorting...");

Iterator<String> iterator =productList.iterator();

**while**(iterator.hasNext()){

System.***out***.println(iterator.next());

}

Arrays.*sort*(products);

System.***out***.println("After sorting...");

**for**(String s:products){

System.***out***.println(s);

}

}

**private** **static** **void** getProductNames(List<String> productList) {

**while**(**true**){

System.***out***.println("Enter product name: ");

productList.add(*scanner*.nextLine());

System.***out***.println("To continue,type yes");

String choice = *scanner*.nextLine();

**if**(!choice.equalsIgnoreCase("yes")){

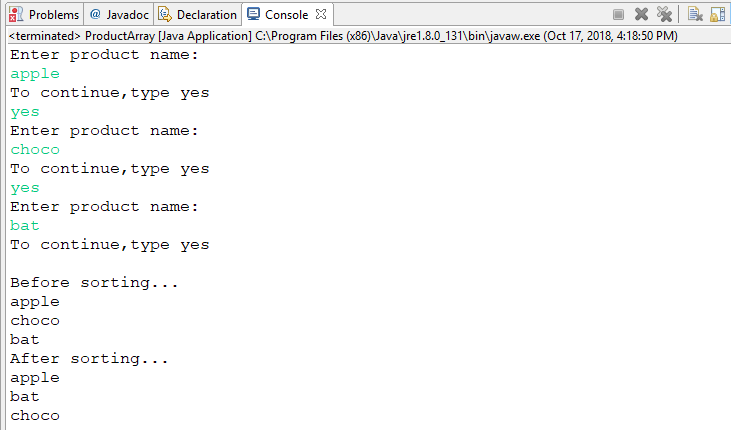
**break**;

}

}

}

}



7.3: Modify the lab assignment 5.1 to accept multiple employee details and store all employee objects in a HashMap. The functionalities need to be implemented are:

1. Add employee details to HashMap.
2. Accept insurance scheme from user and display employee details based on Insurance scheme
3. Delete an employee details from map.
4. Sort the employee details based on salary and display it.

**EmployeeHashMap.java**

**package** com.cg.eis.pl;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Comparator;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Scanner;

**import** com.cg.eis.bean.Employee;

**import** com.cg.eis.exception.EmployeeInsuranceSchemeException;

**public** **class** EmployeeHashMap {

**private** **static** Scanner *scanner*=**new** Scanner(System.***in***);

**static** Map<Long,Employee> *employeeMap*=**new** HashMap<>();

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

*addEmployees*(*employeeMap*);

*showEmployees*(*employeeMap*);

System.***out***.println("----------");

**boolean** bool2=*deleteEmployee*(*employeeMap*);

//System.out.println(bool2);

**if**(bool2==**true**){

System.***out***.println("Your required employee is deleted:");

System.***out***.println("Details of the employees after deleting...");

*showEmployees*(*employeeMap*);

System.***out***.println("-----------");

}**else**{

System.***out***.println("You need to enter the id of an existing employee..");

System.***out***.println("-----------");

}

*acceptInsuranceScheme*(*employeeMap*);

*sortEmployees*(*employeeMap*);

}

**private** **static** **void** sortEmployees(Map<Long, Employee> employeeMap2){

List<Map.Entry<Long,Employee>> entryList=**new** ArrayList<>(employeeMap2.entrySet());

Collections.*sort*(entryList,**new** Comparator<Map.Entry<Long,Employee>>(){

**public** **int** compare(Map.Entry<Long, Employee> e1,Map.Entry<Long,Employee> e2){

**return** e1.getValue().getSalary().compareTo(e2.getValue().getSalary());

}

});

System.***out***.println("After sorting the employees according to their salary..");

System.***out***.println(entryList);

System.***out***.println("------------");

}

**private** **static** **void** acceptInsuranceScheme(Map<Long, Employee> employeeMap3) {

System.***out***.println("Enter the insurance scheme to display the list of employees under that scheme...");

String temp\_insurance=*scanner*.next();

Iterator<Employee> iterator = employeeMap3.values().iterator();

**while**(iterator.hasNext()){

Employee e=iterator.next();

**if**(e.getInsuranceScheme().equals(temp\_insurance)){

System.***out***.println("Employee id: "+e.getId());

System.***out***.println("Employee name: "+e.getName());

System.***out***.println("Employee designation: "+e.getDesignation());

System.***out***.println("Employee salary: "+e.getSalary());

System.***out***.println("Employee insurance scheme: "+e.getInsuranceScheme());

System.***out***.println("----------");

}

}

}

**private** **static** **boolean** deleteEmployee(Map<Long, Employee> employeeMap2) {

System.***out***.println("Enter the id of an employee you want to delete:");

Long del\_id=*scanner*.nextLong();

Iterator<Long> iterator = employeeMap2.keySet().iterator();

**while**(iterator.hasNext()){

Long temp\_id=iterator.next();

**if**(temp\_id==del\_id){

iterator.remove();

**return** **true**;

}

}

**return** **false**;

}

**private** **static** **void** addEmployees(Map<Long, Employee> employeeMap) {

System.***out***.println("Enter number of employees you want to enter:");

**try**{

**int** numberOfEmployees = *scanner*.nextInt();

Employee employee[]=**new** Employee[numberOfEmployees];

**for**(**int** i=0;i<numberOfEmployees;i++){

System.***out***.println("Enter employee " +(i+1)+ " name");

String name=*scanner*.next();

System.***out***.println("Enter the employee "+(i+1)+ " id");

Long id =*scanner*.nextLong();

System.***out***.println("Enter employee "+(i+1)+ " designation");

String designation=*scanner*.next();

System.***out***.println("Enter employee "+(i+1)+ " salary");

Double salary=*scanner*.nextDouble();

System.***out***.println("Enter employee "+(i+1)+ " insurance scheme");

String insuranceScheme=*scanner*.next();

employee[i]=**new** Employee(id,name,salary,designation,insuranceScheme);

**try**{

**boolean** bool=*insuranceSchemeCheck*(employee[i]);

**if**(**true**){

employeeMap.put(id, employee[i]);

System.***out***.println("Data entered for one employee");

}

}**catch**(Exception e){

System.***out***.println(e);

}

}

}**catch**(Exception e){

System.***out***.println("Enter integer"+e);

}

}

**private** **static** **boolean** insuranceSchemeCheck(Employee employee) **throws** EmployeeInsuranceSchemeException {

**if**((employee.getSalary()>5000 && employee.getSalary()<20000) && (employee.getDesignation().equalsIgnoreCase("System Associate"))&& (employee.getInsuranceScheme().equalsIgnoreCase("A")))

{

**return** **true**;

}

**else** **if**((employee.getSalary()>=20000) && (employee.getSalary()<40000) && (employee.getDesignation().equalsIgnoreCase("Programmer"))&& (employee.getInsuranceScheme().equalsIgnoreCase("B")))

{

**return** **true**;

}

**else** **if**((employee.getSalary()>=40000 )&& (employee.getDesignation().equalsIgnoreCase("Manager"))&&(employee.getInsuranceScheme().equalsIgnoreCase("C")))

{

**return** **true**;

}

**else** **if**((employee.getSalary()<=5000) && (employee.getDesignation().equalsIgnoreCase("Clerk"))&&(employee.getInsuranceScheme().equalsIgnoreCase("No Scheme")))

{

**return** **true**;

}

**else**{

**throw** **new** EmployeeInsuranceSchemeException();

}

}

**private** **static** **void** showEmployees(Map<Long, Employee> employeeMap) {

**for**(Map.Entry<Long,Employee> e:employeeMap.entrySet()){

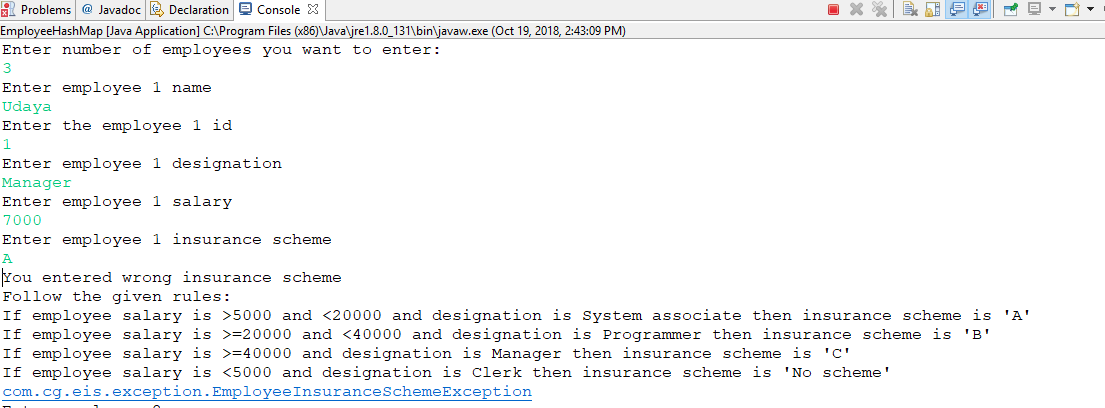
System.***out***.println(e.getKey()+":"+e.getValue());

}

}

}

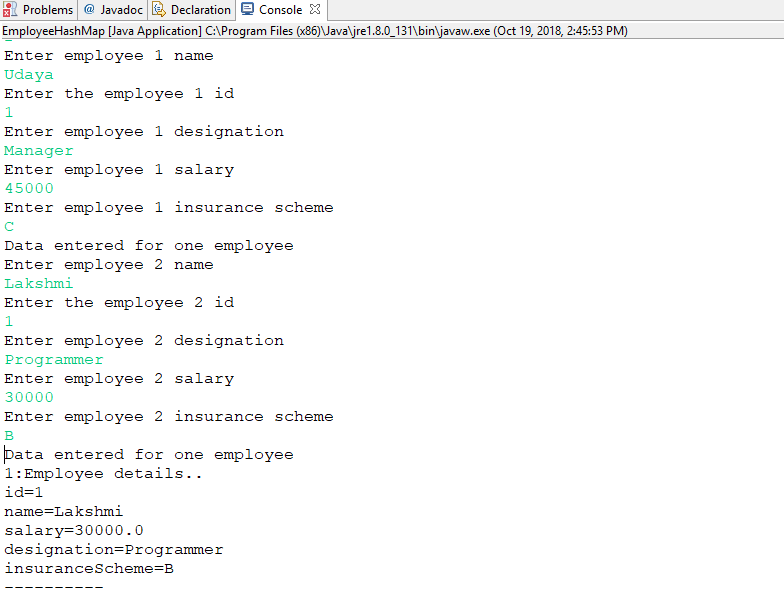
**Case 1:if user entered wrong information**



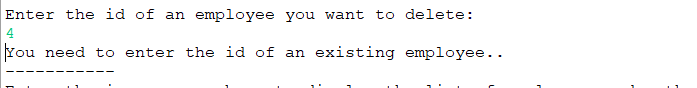
Case 2: After adding details of employees



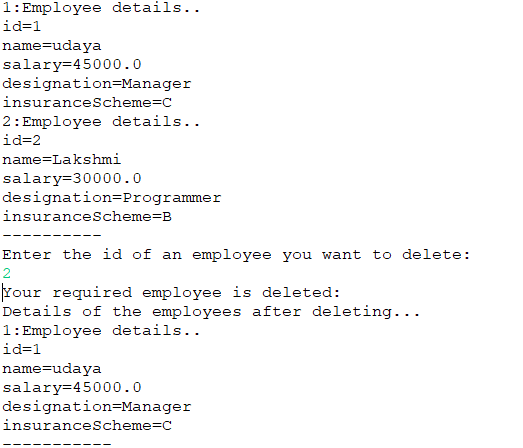
Case 3: if we enter same employee id for more than one employee



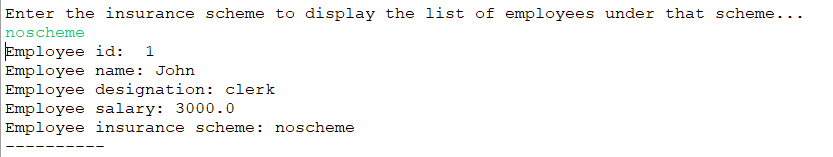
Case4: Deleting an employee who is not existing



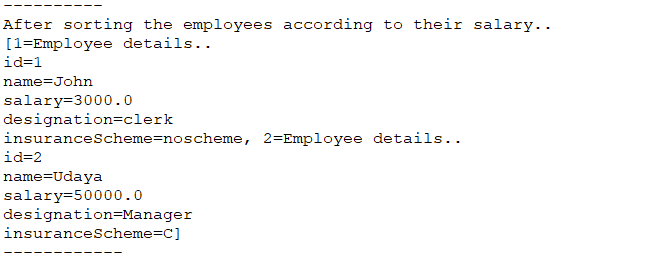
Case5: Deleting an existing employee:



Case 6: Getting details of employees based on Insurance Schemes.



Case 7: After sorting according to their salary:



**8.Files I/O**

8.1: Write a program to read content from file, reverse the content and write the reversed content to the file. (Use Reader and Writer APIs).

**package** com.capgemini.trg.ui;

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.LineNumberReader;

**import** java.io.PrintWriter;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ReversedContentToFile {

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

*writeToFile*();

*readFromFile*();

*reverseFile*();

}

**private** **static** **void** reverseFile() {

File file=**new** File("C:\\Data\\lines.txt");

String s;

String reverse="";

**if**(file.exists() && file.canRead()){

**try**(

FileReader fr=**new** FileReader(file);

BufferedReader br=**new** BufferedReader(fr);

LineNumberReader lnr=**new** LineNumberReader(br);

){

List<String> tmp=**new** ArrayList<>();

s=br.readLine();

**while**(s!=**null**){

tmp.add(s);

s=br.readLine();

}

System.***out***.println("Reverse order of lines:");

**for**(**int** i=tmp.size()-1;i>=0;i--){

System.***out***.println(tmp.get(i));

String str=tmp.get(i);

**for**(**int** j=str.length()-1;j>=0;j--){

reverse=reverse+str.charAt(j);

}

}

System.***out***.println("Reverse order of data");

System.***out***.println(reverse);

FileWriter fw=**new** FileWriter("C:\\data\\rev.txt");

BufferedWriter bw=**new** BufferedWriter(fw);

bw.write(reverse);

bw.close();

fw.close();

System.***out***.println("Success");

}**catch**(IOException e){

e.printStackTrace();

}**catch**(Exception e){

e.printStackTrace();

}

}**else**{

System.***out***.println("Unable to open the file");

}

}

**private** **static** **void** readFromFile() {

File file=**new** File("C:\\Data\\lines.txt");

**if**(file.exists() && file.canRead()){

**try**(

FileReader fr=**new** FileReader(file);

BufferedReader br=**new** BufferedReader(fr);

LineNumberReader lnr=**new** LineNumberReader(br);

){

String line;

**while**((line=lnr.readLine())!=**null**){

System.***out***.println(lnr.getLineNumber()+"."+line);

}

}**catch**(IOException e){

e.printStackTrace();

}**catch**(Exception e){

e.printStackTrace();

}

}**else**{

System.***out***.println("Unable to open the file");

}

}

**private** **static** **void** writeToFile() {

**try**(

FileWriter fw=**new** FileWriter("C:\\Data\\lines.txt");

BufferedWriter bw=**new** BufferedWriter(fw);

PrintWriter pw=**new** PrintWriter(bw);

InputStreamReader isr=**new** InputStreamReader(System.***in***);

BufferedReader br=**new** BufferedReader(isr);

){

System.***out***.println("Enter number of lines:");

**int** size=Integer.*parseInt*(br.readLine());

**for**(**int** i=0;i<size;i++){

System.***out***.println("Enter Line:"+(i+1));

pw.print(br.readLine());

pw.println();

}

}**catch**(IOException e){

e.printStackTrace();

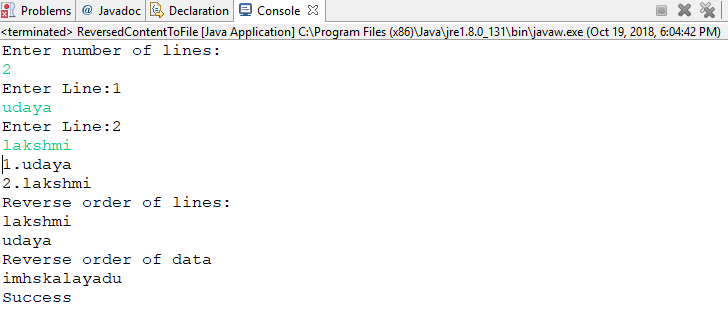
}**catch**(Exception e){

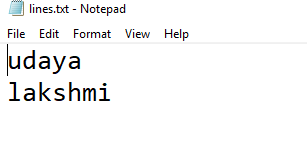
e.printStackTrace();

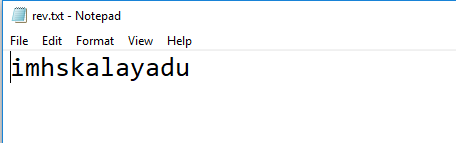
}

}

}







8.2: Create a file named as “numbers.txt” which should contain numbers from 0 to 10 delimited by comma. Write a program to read data from numbers.txt using Scanner class API and display only even numbers in the console.

package com.capgemini.trg.ui;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class EvennumFromFile {

public static void main(String[] args) {

FileWriter fw;

writeNumbersToFile();

readNumberFromFile();

}

private static void readNumberFromFile() {

File file=new File("C:\\data\\number.txt");

if(file.exists() && file.canRead()){

try {

Scanner sc=new Scanner(file);

sc.useDelimiter(",");

System.out.println(“Even numbers from file:”);

while(sc.hasNext()){

Integer num=sc.nextInt();

if(num%2==0){

System.out.println(num);

}

}

sc.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

}

private static void writeNumbersToFile() {

FileWriter fw;

try {

fw = new FileWriter("C:\\data\\number.txt");

BufferedWriter bw=new BufferedWriter(fw);

bw.write("0,1,2,3,4,5,6,7,8,9,10");

bw.close();

fw.close();

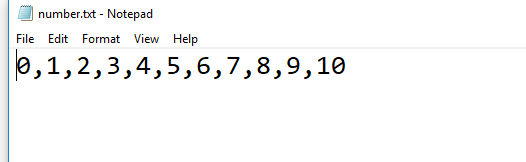
} catch (IOException e) {

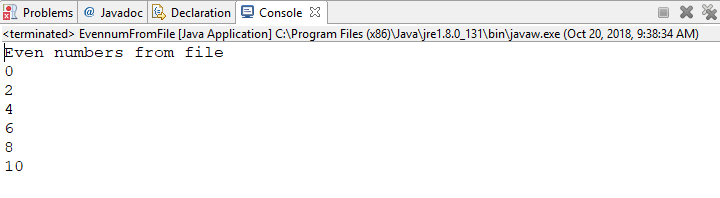
e.printStackTrace();

}

}

}





8.3: Enhance the lab assignment 6.3 by adding functionality in service class to write employee objects into a File. Also read employee details from file and display the same in console. Analyze the output of the program.

**package** com.cg.eis.service;

**import** java.io.EOFException;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**import** java.util.Arrays;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Scanner;

**import** com.cg.eis.bean.Employee;

**import** com.cg.eis.exception.EmployeeException;

**import** com.cg.eis.pl.User;

**public** **class** EmployeeObjectsIntoFile {

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

File file =

**new** File("C:\\data\\employees.txt");

List<Employee> employeeList = *populateEmployeeList*();

*writeEmployees*(file,employeeList);

*readFromFile*(file);

}

**private** **static** **void** readFromFile(File file) {

**if**(file.exists()&&file.canRead()){

**try**(

FileInputStream fis = **new** FileInputStream(file);

ObjectInputStream ois = **new** ObjectInputStream(fis);

){

**while**(**true**){

Employee employee = (Employee)ois.readObject();

System.***out***.println(employee);

}

}**catch**(EOFException e){

System.***out***.println("End of file");

}**catch**(IOException e){

e.printStackTrace();

}**catch**(Exception e){

e.printStackTrace();

}

}**else**{

System.***out***.println("Unable to open the file");

}

}

**private** **static** List<Employee> populateEmployeeList() {

Scanner scanner=**new** Scanner(System.***in***);

System.***out***.println("Enter employee name");

String name=scanner.nextLine();

System.***out***.println("Enter employee designation");

String designation=scanner.nextLine();

System.***out***.println("Enter the employee id");

Long id =scanner.nextLong();

System.***out***.println("Enter employee salary");

Double salary=scanner.nextDouble();

User user=**new** User();

Employee employee1;

**try** {

employee1 = **new** Employee(id,name,salary,designation);

String insuranceScheme=user.employeeInsuranceSchemeOffers(employee1);

Employee employee2;

employee2 = **new** Employee(id,name,salary,designation,insuranceScheme);

//System.out.println(employee2);

List empList=Arrays.*asList*(employee2);

**return** empList;

} **catch** (EmployeeException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}**finally**{

scanner.close();

}

**return** **null**;

}

**private** **static** **void** writeEmployees(File file, List<Employee> employeeList) {

**try**(

FileOutputStream fos = **new** FileOutputStream(file);

ObjectOutputStream oos = **new** ObjectOutputStream(fos);

){

Iterator<Employee> iterator = employeeList.iterator();

**while**(iterator.hasNext()){

oos.writeObject(iterator.next());

}oos.flush();

System.***out***.println("Employee Serialization Completed");

}**catch**(IOException e){

}**catch**(Exception e){

}

}

}

Employee.java

package com.cg.eis.bean;

import java.io.Serializable;

import com.cg.eis.exception.EmployeeException;

public class Employee implements Serializable{

private static final long serialVersionUID = 1L;

private Long id;

private String name;

private Double salary;

private String designation;

private String insuranceScheme;

public Employee() {

super();

}

public Employee(Long id, String name, Double salary, String designation)throws EmployeeException {

super();

this.id = id;

this.name = name;

if(salary<3000.00){

throw new EmployeeException();

}

this.salary=salary;

this.designation = designation;

}

public Employee(Long id, String name, Double salary, String designation,

String insuranceScheme) /\*throws EmployeeException ,EmployeeInsuranceSchemeException\*/{

super();

this.id = id;

this.name = name;

this.salary = salary;

this.designation = designation;

this.insuranceScheme = insuranceScheme;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Double getSalary() {

return salary;

}

public void setSalary(Double salary) {

this.salary = salary;

}

public String getDesignation() {

return designation;

}

public void setDesignation(String designation) {

this.designation = designation;

}

public String getInsuranceScheme() {

return insuranceScheme;

}

public void setInsuranceScheme(String insuranceScheme) {

this.insuranceScheme = insuranceScheme;

}

@Override

public String toString() {

return "Employee details.."+"\n"+

"id=" + getId() +"\n"+

"name=" + getName() +"\n"+

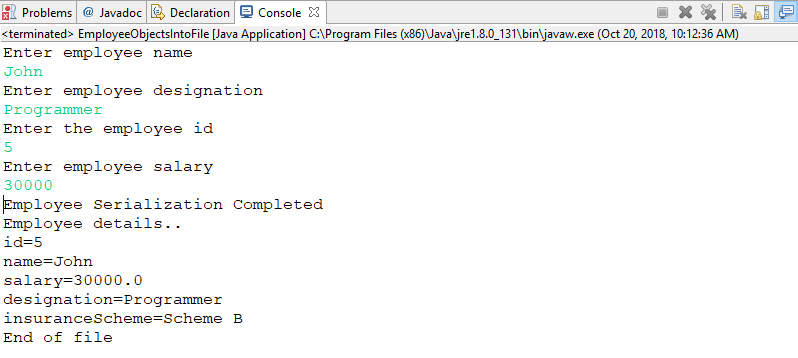
"salary=" + getSalary()+"\n"+

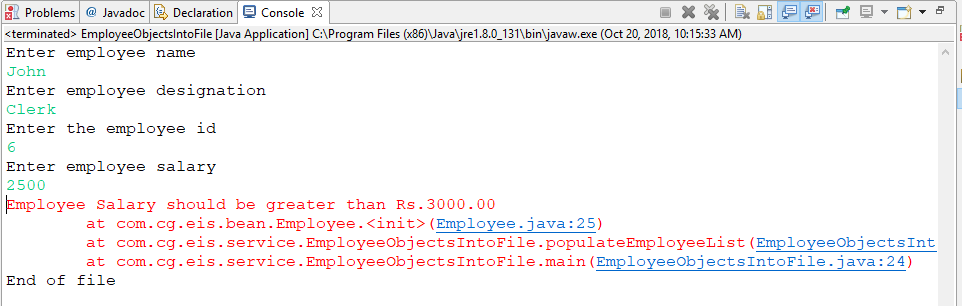
"designation=" + getDesignation() +"\n"+

"insuranceScheme="+ getInsuranceScheme();

}

}





**Introduction of JUnit:**

## Writing JUnit tests

Consider the following Java program. Write tests for testing various methods in the class

**package** com.capgemini.trg.service;

**public** **class** DateExample {

**int** intDay, intMonth, intYear;

**public** DateExample() {

**super**();

}

**public** DateExample(**int** intDay, **int** intMonth, **int** intYear) {

**super**();

**this**.intDay = intDay;

**this**.intMonth = intMonth;

**this**.intYear = intYear;

}

**public** **int** getIntDay() {

**return** intDay;

}

**public** **void** setIntDay(**int** intDay) {

**this**.intDay = intDay;

}

**public** **int** getIntMonth() {

**return** intMonth;

}

**public** **void** setIntMonth(**int** intMonth) {

**this**.intMonth = intMonth;

}

**public** **int** getIntYear() {

**return** intYear;

}

**public** **void** setIntYear(**int** intYear) {

**this**.intYear = intYear;

}

@Override

**public** String toString() {

**return** "DateExample [intDay=" + intDay + ", intMonth=" + intMonth

+ ", intYear=" + intYear + "]";

}

}

DateExampleTest.java

package com.capgemini.trg.service;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertTrue;

import static org.junit.Assert.fail;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.time.format.FormatStyle;

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Test;

public class DateExampleTest {

@BeforeClass

public static void setUpBeforeClass() throws Exception {

System.out.println("Initialization common for all test methods.");

}

@AfterClass

public static void tearDownAfterClass() throws Exception {

System.out.println("Clean up after executing all test methods..");

}

@Before

public void setUp() throws Exception {

System.out.println("This executes before each test method");

}

@After

public void tearDown() throws Exception {

System.out.println("This executes after eash test method");

}

@Test

public void testGetIntDay() {

assertEquals(20,new DateExample(20,10,2018).getIntDay());

}

@Test

public void testGetIntMonth() {

assertEquals(10,new DateExample(20,10,2018).getIntMonth());

}

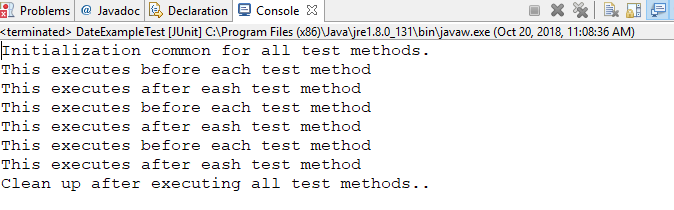
@Test

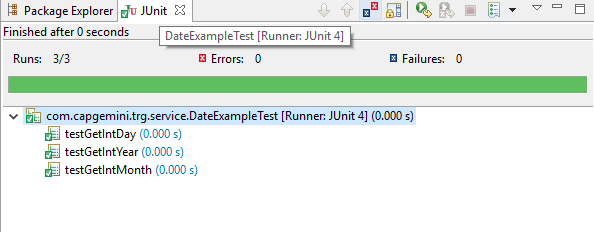
public void testGetIntYear() {

assertEquals(2018,new DateExample(20,10,2018).getIntYear());

}

}





**9.2.2:** Consider the lab assignment 6.3 from Exception Handling Lab. Create a new class ExceptionCheck.javawhich handles an exception. Write a test case to verify if the exception is being handled correctly.

EmployeeTest.java

package com.capgemini.trg.service;

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Test;

public class EmployeeTest {

@BeforeClass

public static void setUpBeforeClass() throws Exception {

}

@AfterClass

public static void tearDownAfterClass() throws Exception {

}

@Before

public void setUp() throws Exception {

}

@After

public void tearDown() throws Exception {

}

@Test(expected=EmployeeException.class)

public void testEmployeeException() {

try {

Employee employee=new Employee(100L,"John",2000.00,"Clerk");

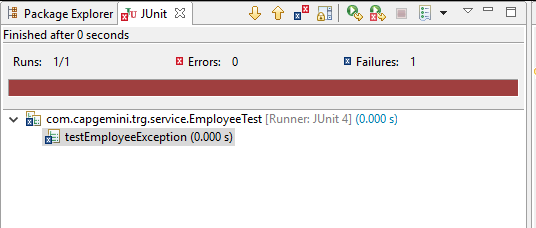
} catch (EmployeeException e) {

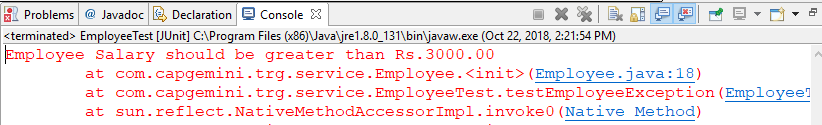
e.printStackTrace();

}

}

}





**10.Properties and JDBC**

10.1: Write a program to store a person details in a properties file named as “PersonProps.properties” and also do the following tasks:

1. Read data from properties file, load the data into Properties object and display the data in the console.

**PersonDetailsProperties.java**

**package** com.capgemini.trg.utility;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.util.Properties;

**public** **class** PersonDetailsProperties {

**private** **static** Properties *properties*=**new** Properties();

**static**{

*properties*.setProperty("FirstName", "Udaya");

*properties*.setProperty("LastName", "Lakshmi");

*properties*.setProperty("Gender", "Female");

*properties*.setProperty("Age", "21");

*properties*.setProperty("Id", "382115007925");

}

**public** **static** **void** saveProperties(String fileName){

**try**(

FileOutputStream fos=**new** FileOutputStream(fileName);

){

*properties*.store(fos,"PersonDetails Property file");

System.***out***.println("Property file created");

}**catch**(IOException e){

e.printStackTrace();

}**catch**(Exception e){

e.printStackTrace();

}

}

**public** **static** Properties loadProperties(String fileName){

File file=**new** File(fileName);

**if**(file.exists() && file.canRead()){

**try**(

FileInputStream fis=**new** FileInputStream(fileName);

){

Properties properties=**new** Properties();

properties.load(fis);

**return** properties;

}**catch**(IOException e){

e.printStackTrace();

}**catch**(Exception e){

e.printStackTrace();

}

**return** **null**;

}

**else**{

System.***out***.println("Unable to open the file");

**return** **null**;

}

}

}

**PersonDetailsPropertiesTester.java**

**package** com.capgemini.trg.ui;

**import** java.util.Properties;

**import** com.capgemini.trg.utility.PersonDetailsProperties;

**public** **class** PersonDetailsPropertiesTester {

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

String fileName="C:\\data\\PersonProps.properties";

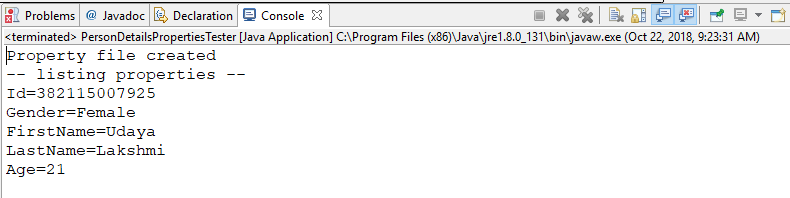
PersonDetailsProperties.*saveProperties*(fileName);

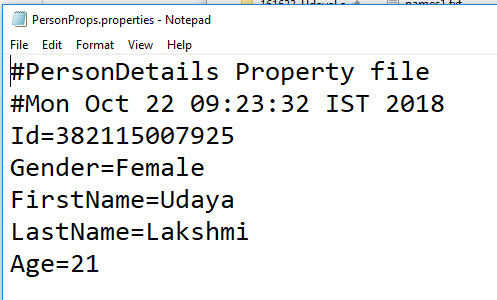
Properties properties=PersonDetailsProperties.*loadProperties*(fileName);

properties.list(System.***out***);

}

}





b) Read data from properties file(using getProperties method) and print data in the console.

PersonDetailsPropertiesTester.java

**package** com.capgemini.trg.ui;

**import** java.util.Properties;

**import** com.capgemini.trg.utility.PersonDetailsProperties;

**public** **class** PersonDetailsPropertiesTester {

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

String fileName="C:\\data\\PersonProps.properties";

PersonDetailsProperties.*saveProperties*(fileName);

Properties properties=PersonDetailsProperties.*loadProperties*(fileName);

//properties.list(System.out);

System.***out***.println("FirstName: "+properties.getProperty("FirstName"));

System.***out***.println("LastName: "+properties.getProperty("LastName"));

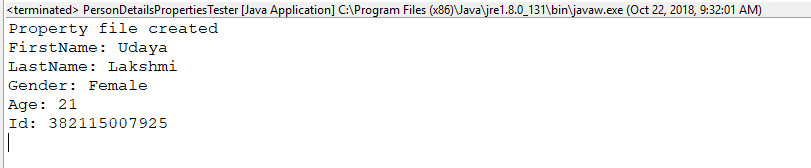
System.***out***.println("Gender: "+properties.getProperty("Gender"));

System.***out***.println("Age: "+properties.getProperty("Age"));

System.***out***.println("Id: "+properties.getProperty("Id"));

}

}



10.2: Extend the assignment 7.3 by persisting data into database instead of hashmap and display/delete data from database. Use DriverManager for connecting to the database.