Velammal College of Engineering and Technology, Madurai-625009
(Autonomous)
Department of Computer Science and Engineering
Assignment II

Degree: B.E/CSE Year/Semester/section : II/III/B
Course Code-Title: 21CS203/Object Oriented Programming Batch: 2021-25
Name of the Instructor: J.Shanthalakshmi Revathy
Announcement Date : 1.10.22
Submission Date: 10.10.22
Total Marks: 20 Relevant COs : CO2-K3

Questions:

Create a TaxReturn class with fields that hold a taxpayer's Social Security number, last name, first name, street address, city, state, zip code, annual income, marital status, and tax liability. Include a constructor that requires arguments that provide values for all the fields other than the tax liability. The constructor calculates the tax liability based on annual income and the percentages in the following table.

In the TaxReturn class, also include a display method that displays all the TaxReturn data. Save the file as TaxReturn.java

Create an application that prompts a user for the data needed to create a TaxReturn. Continue to prompt the user for data as long as any of the following are true:

 The Social Security number is not in the correct format, with digits and dashes in the appropriate positions—for example, 999-99-9999.

 The zip code is not five digits.

 The marital status does not begin with one of the following: "S", "s", "M", or "m".

 The annual income is negative.

After all the input data is correct, create a TaxReturn object and then display its values. Save the file as PrepareTax.java

ALGORITHM:
STEP 1: start
STEP 2: Create a TaxReturn class with fields that hold a taxpayer's Social Security number, last name, first name, street address, city, state, zip code, annual income, marital status, and tax liability
STEP 3: create The constructor calculates the tax liability based on annual income and the percentages
3.1: if income is in between 0-20,000 then the tax liability for single is 15% and married is 14%
3.2: if 20,001-50,000 then the tax liability for single is 22% and married is 20%
3.3: if 50,001 and over then the tax liability for single is 30% and married is 28%
STEP 4: create a methods to display the all the tax return data.
STEP 5: Create an application that prompts a user for the data needed to create a TaxReturn.
STEP 6: After getting all the input data is correct, create a TaxReturn object and then display its values.
Save the file as PrepareTax.java
STEP 7: stop.
CODE:

CLASS FILE:TAXRETURN

```
package preparetax;
class taxreturn
{
String Social_Security_Number;
String last_name;
String first_Name;
String Address;
String city;
String state;
String zip_code;
double annual_Income;
String marital_Status;
```

```
double tax_liability;
public TAXreturn(String ssn, String last, String first, String add, String ct,
String st, String zip,double income, String status)
{
this.Social_Security_Number=ssn;
this.last_name=last;
this.first_Name=first;
this.Address=add;
this.city=ct;
this.state=st;
this.zip_code=zip;
this.annual_Income=income;
this.marital_Status=status;
}
void calculateTax()
{
if(annual_Income>0 &&annual_Income<=20000 )
{
if(this.marital_Status.equalsIgnoreCase("Single"))
{
tax_liability=this.annual_Income*15/100;
}
else
{
tax_liability=this.annual_Income*14/100;
}
```

```
}
else if(annual_Income>=20001 &&annual_Income<=50000 )
{
if(this.marital_Status.equalsIgnoreCase("Single"))
{
tax_liability=this.annual_Income*22/100;
}
else
{
tax_liability=this.annual_Income*20/100;
}
}
else if(annual_Income>=50001)
{
if(this.marital_Status.equalsIgnoreCase("Single"))
{
tax_liability=this.annual_Income*30/100;
}
else
{
tax_liability=this.annual_Income*28/100;
}
}
}
public void display()
{
```

```
System.out.println("Social Security Number: "+this.Social_Security_Number);
System.out.println("Last Name: "+this.last_name);
System.out.println("First Name: "+this.first_Name);
System.out.println("Street address: "+this.Address);
System.out.println("City: "+this.city);
System.out.println("State: "+this.state);
System.out.println("Zipcode: "+this.zip_code);
System.out.println("Annual Income: "+this.annual_Income);
System.out.println("Marital Status: "+this.marital_Status);
System.out.println("Tax liability: "+this.tax_liability);
}
}
```

CLASS FILE: PREPARETAX

```
package preparetax;
import java.util.Scanner;
public class preparetax
{
public static void main(String[] args)
{
Scanner input=new Scanner(System.in);
String ssn="";
int flag=0;
while(flag==0){
System.out.println("Enter Social Security Number: ");
ssn=input.nextLine();
if(ssn.length()==11 && ssn.charAt(3)=='-' && ssn.charAt(6)=='-')
{
flag=1;
break;
}
else
{
System.out.println("Social Security Number should be in the form of 999-99-9999 ");
}
}
System.out.println("Last Name: ");
```

```
String last=input.nextLine();
System.out.println("First Name: ");
String first=input.nextLine();
System.out.println("Marital Status: ");
String status=input.nextLine();
System.out.println("Street address: ");
String add=input.nextLine();
System.out.println("City: ");
String ct=input.nextLine();
System.out.println("State: ");
String st=input.nextLine();
flag=0;
String zip="";
while(flag==0){
System.out.println("Enter the Zipcode: ");
zip=input.nextLine();
if(zip.length()==5)
{
flag=1;
break;
}
else
{
System.out.println("Zipcode length must be five");
}
}
flag=0;
double income=0;
while(flag==0){
System.out.println("Enter Annual Income: ");
income=input.nextDouble();
if(income>0)
{
flag=1;
break;
}
else
{
System.out.println("Annual income should be greater than zero");
}
}
taxreturn taxreturn =new taxreturn(ssn,last,first,add,ct,st,zip,income,status);
taxreturn.calculateTax();
taxreturn.display();
}
}
```

OUTPUT:

Enter Social Security Number:

123-456-7890

Last Name:

A

First Name:

Rithikha

Marital Status:

Single

Street address:

abc street

City:

Madurai

State:

Tamilnadu

Enter the Zipcode:

12345

Enter Annual Income:

1000000

Social Security Number: 123-456-7890

Last Name: A

First Name: Rithikha

Street address: abc street

City:Madurai

State: Tamilnadu

Zipcode: 12345

Annual Income: 1000000.0

Marital Status: Single

Tax liability: 300000.0

BUILD SUCCESSFUL (total time: 2 minutes 56 seconds)