# CS500-01 Fundamentals of Programming

# Assignment 4

Tharinda Embuldeniya

S01996041

**Question 1**:

**Implement the class Parallelogram to represent a Parallelogram.**

**The class contains:**

**-         double data fields named base, side and height. The default value of both is 1**

**-         A default constructor with no-arg.**

**-         A method named setBase()**

**-        A method named setSide()**

**-         A method named setHeight()**

**-         A method named getBase()**

**-          A method named getSide()**

**-         A method named getHeight()**

**-         A method named getArea()**

**-         A method named getPerimeter()**

**Write a test program that creates two Parallelogram objects P1 and P2.**

**Set the base, side  and height of P1 and P2 to some values**

**Get the area and perimeter of P1**

**Get the base, side, height, area and perimeter of P2.**

*Code*

**public** **class** Parallelogram {

//double data fields named base, side and height.

**private** **double** base;

**private** **double** side;

**private** **double** height;

//Constructor with default value set to 1

Parallelogram(){

base =1.0;

height=1.0;

side = 1.0;

}

//Constructor with with no-args

Parallelogram(**double** newBase, **double** newSide, **double** newHeight){

base = newBase;

height= newHeight;

side = newSide;

}

//setters

// A method named setBase()

**public** **void** setBase(**double** newBase) {

base = newBase;

}

//A method named setSide()

**public** **void** setSide(**double** newSide) {

side = newSide;

}

// A method named setHeight()

**public** **void** setHeight(**double** newHeight) {

height = newHeight;

}

//getters

// A method named getBase()

**public** **double** getBase() {

**return** base;

}

// A method named getSide()

**public** **double** getSide() {

**return** side;

}

//A method named getHeight()

**public** **double** getHeight() {

**return** height;

}

// A method named getArea()

**public** **double** getArea() {

**return** base \*height;

}

// A method named getPerimeter()

**public** **double** getPerimeter() {

**return** 2 \*(side + base );

}

}

**Main Class**

**public** **class** Main\_Parallelogram {

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

//define two objects with default parameters

Parallelogram p1 = **new** Parallelogram ();

Parallelogram p2 = **new** Parallelogram ();

//Set the base, side and height of P1

p1.setBase(6.9);

p1.setHeight(7.2);

p1.setSide(8.2);

//Set the base, side and height of P2

p2.setBase(9.1);

p2.setHeight(7.7);

p2.setSide(4.2);

//Get the area and perimeter of P1

System.***out***.println("\nP1");

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

System.***out***.println("Area:\t" + p1.getArea());

System.***out***.println("Perimeter:\t" + p1.getPerimeter());

//Get the base, side, height, area and perimeter of P2.

System.***out***.println("\nP2");

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

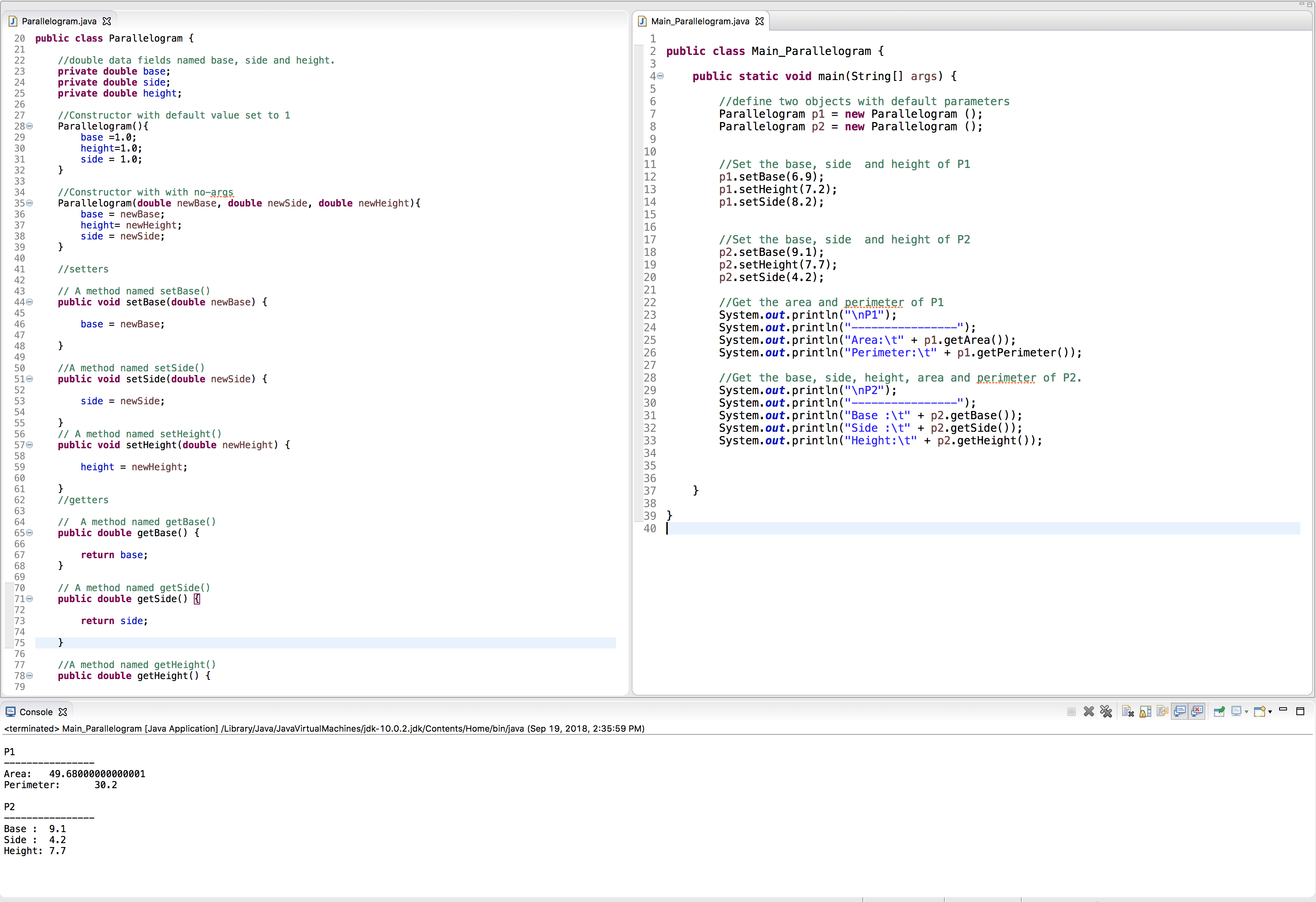
System.***out***.println("Base :\t" + p2.getBase());

System.***out***.println("Side :\t" + p2.getSide());

System.***out***.println("Height:\t" + p2.getHeight());

}

}



**Question 2:**

**Implement the class Voter to represent a voting member of the community.**

**The class contains:**

**-         data field name**

**-         data filed address**

**-         data field age**

**-         data field tel. number**

**-         data field voting ID number**

**-          A default constructor with no-arg that sets all data fields to either 0, or space**

**-         A constructor that creates an object with name, address, age, tel and ID passed**

**-         A method named setName()**

**-         A method named setAddress()**

**-         A method named setAge() – the welcome() method will call it**

**-         A method named setTel()**

**-         A method named setID()**

**-         A method named getName()**

**-         A method named getAddress()**

**-         A method named getAge()**

**-         A method named getTel()**

**-         A method named getID()**

**-         A Boolean method checkAge() will return true if the voter’s age is greater than or equal to 18; otherwise will return false.**

**-         A method named Welcome() that will ask the voter for his/her age, call checkAge() and if it returns true will display a welcome message and setAge(); otherwise will display a message asking them to return when they are 18**

**-**

**Write a test program that creates two objects V1 and V2.**

**Create V1 using the default constructor.**

**Set all data fields using the set methods and welcome()**

**Create V2 using the constructor passing all arguments**

**Get all data fields using the get methods.**

**Test it using your own created data.**

Code

**public** **class** Voter {

// data field name

**private** String name;

// data field address

**private** String address;

// data field age

**private** **int** age;

// data field tel. number

**private** String tel;

// data field voting ID number

**private** **int** id;

//A default constructor with no-arg that sets all data fields to either 0, or space

Voter(){

name = " ";

address= " ";

age = 0;

tel = " ";

id = 0;

}

//A constructor that creates an object with name, address, age, tel and ID passed

Voter(String newName, String newAddress ,**int** newAge, String newTel, **int** newId){

name = newName;

address = newAddress;

age = newAge;

tel = newTel;

id = newId;

}

//setters

// method named setName()

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

name = newName;

}

// A method named setAddress()

**public** **void** setAddress (String newAddress) {

address = newAddress;

}

// A method named setAge()

**public** **void** setAge(**int** newAge) {

age = newAge;

}

//A method named setTel()

**public** **void** setTel (String newTel) {

tel = newTel;

}

//A method named setID()

**public** **void** setId(**int** newId) {

id = newId;

}

//A method named getName()

**public** String getName() {

**return** name;

}

// A method named getAddress()

**public** String getAddress() {

**return** address;

}

//A method named getAge()

**public** **int** getAge() {

**return** age;

}

//A method named getTel()

**public** String getTel() {

**return** tel;

}

//A method named getID()

**public** **int** getId() {

**return** id;

}

//A Boolean method checkAge() will return true if the voter’s age is greater than or equal to 18; otherwise will return false.

**public** **boolean** checkAge() {

**if** (getAge()>=18) {

**return** **true**;

}

**else** {

**return** **false**;

}

}

// A method named Welcome() that will ask the voter for his/her age, call checkAge() and if it returns true will display a welcome message and setAge(); otherwise will display a message asking them to return when they are 18

**public** **void** welcome(**int** newAge) {

setAge(newAge);

**if** (checkAge()) {

System.***out***.println("\nWelcome To Vote");

}

**else** {

System.***out***.println("\nPlease come again after 18 years olds");

}

}

}

Main Class

**public** **class** Main\_Voter {

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

//define object V1

//Create V1 using the default constructor.

Voter v1 = **new** Voter();

//Set all data fields using the set methods and welcome()

v1.setName("James");

v1.setAddress("348 William Street , HighValley, 54113 ");

v1.setAge(45);

v1.setTel("200-542-1234");

v1.setId(34214);

//ask the user for the age

v1.welcome(45);

//define object V2

//Create V2 using the constructor passing all arguments

Voter v2 = **new** Voter("Naomi","71 Hill Street, HighValley, 65312", 17, "200-532-4321",342132);

//print out the output

System.***out***.print("\nName: "+ v2.getName() +

"\nAddress: "+ v2.getAddress() +

"\nAge: "+ v2.getAge() +

"\nTel: "+ v2.getTel()+

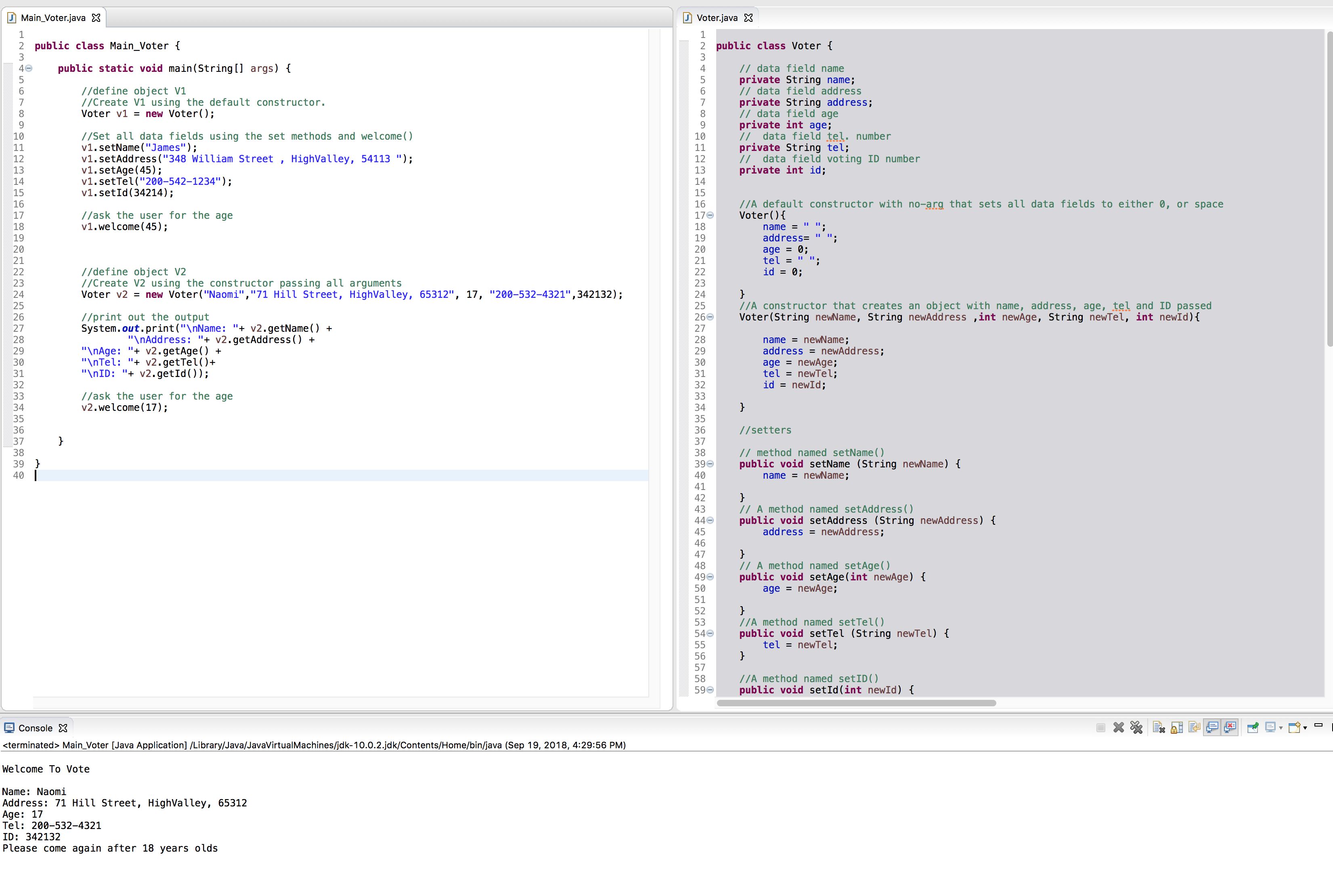
"\nID: "+ v2.getId());

//ask the user for the age

v2.welcome(17);

}

}



**Question 3:**

**Implement the class StackOfDoubles.**

**The class contains:**

**-         data field array of doubles**

**-         data field size**

**-         data field (constant) DEFAULT\_CAPACITY = 10**

**-         a default constructor to construct a stack with the default capacity.**

**-         A constructor to construct a stack with a specified capacity passed.**

**-         a method named empty()**

**-         a method named peek()**

**-         a method named pop() – print what you are popping.**

**-         a method named getSize()**

**-         a method named push()**

**Write a test program that creates two objects S1 with default constructor, S2 with capacity of 20.**

**on S1, push 10, 20, 30 pop once then push 40 50 peek push 60 get size then pop everything until empty.**

**On S2, push 1, 2, 3, 4, 5, 6,peek and get size pop 3 times then push 7, 8, 9, then pop everything until empty.**

Code

StackOfDoubles

//based on book listing 10.8 amend the code

**public** **class** StackOfDoubles {

//data field array of doubles

**private** **double** [] doubles;

// data field size

**private** **int** size;

// data field (constant) DEFAULT\_CAPACITY = 10

**private** **static** **final** **int** ***DEFAULT\_CAPACITY*** =10;

//default constructor to construct a stack with the default capacity

StackOfDoubles(){

**this**(***DEFAULT\_CAPACITY***);

}

// A constructor to construct a stack with a specified capacity passed.

StackOfDoubles(**int** capacity){

doubles = **new** **double** [capacity];

}

// a method named push()

**public** **void** push (**double** value) {

**if**(size >= doubles.length) {

**double** [] temp = **new** **double**[doubles.length\*2];

System.*arraycopy*(doubles, 0, temp, 0, doubles.length);

doubles = temp;

}

doubles[size++] = value;

}

// a method named peek()

**public** **double** peek() {

**return** doubles[size-1];

}

// a method named empty()

**public** **boolean** empty() {

**return** size == 0;

}

// a method named pop()

**public** **double** pop() {

**return** doubles[--size];

}

// getters

//a method named getSize()

**public** **int** getSize() {

**return** size;

}

}

Main Class

**public** **class** Main\_StackOfDoubles {

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

//S1 with default constructor

StackOfDoubles s1 = **new** StackOfDoubles();

s1.push(10);

s1.push(20);

s1.push(30);

s1.pop();

s1.push(40);

s1.push(50);

System.***out***.println("Peek: "+s1.peek());

s1.push(60);

s1.getSize();

**while**(!s1.empty()) {

System.***out***.print (s1.pop() + " ");

}

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

//S2 with capacity of 20.

StackOfDoubles s2 = **new** StackOfDoubles(20);

//push 1, 2, 3, 4, 5, 6 using a loop

**for** (**int** i=1; i<7; i++) {

s2.push(i);

}

System.***out***.println("Peek: "+ s2.peek());

System.***out***.println("Size: "+ s2.getSize());

// pop three times

**for** (**int** i=1; i<4; i++) {

s2.pop();

}

// push 7,8,9 using a loop

**for** (**int** i=7; i<10; i++) {

s2.push(i);

}

//pop everything out until list empty

**while**(!s2.empty()) {

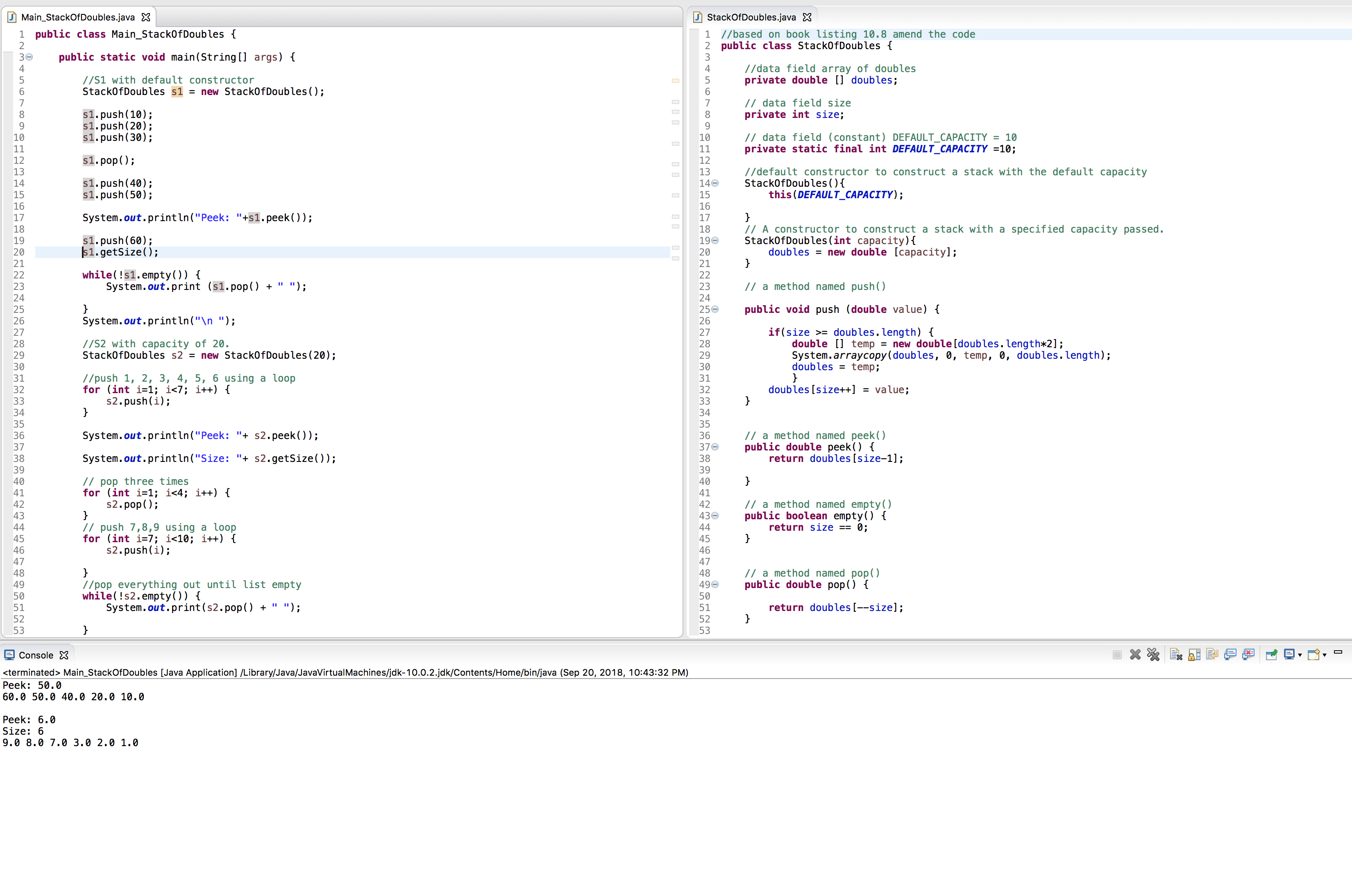
System.***out***.print(s2.pop() + " ");

}

}

}

Screenshot



**Question 4:**

**Implement the class MyMoto.**

**The class contains:**

**-         data field moto of type string**

**-         a default constructor to set data fields to spaces.**

**-         a method named getMoto() to display the moto**

**-         a method named setMoto() to change the moto**

**-         a method named replace to replace a word with another word in the moto**

**-         a method named matches to check if a word matches with a word in the moto**

**-         A method named append to append to the end of the moto**

**-         A method named delete to remove a word from the moto**

**-         A method named insert to insert a word into the moto at a specific location**

**-         a method named length to get the length of the moto**

**Write a test program that creates an object M1.**

**Display a menu similar to the following:**

**Choose 1 to create a new moto**

**Choose 2 to set moto**

**Choose 3 to get moto**

**Choose 4 to replace a word**

**Choose 5 to match a word**

**Choose 6 to append to moto**

**Choose 7 to delete a word**

**Choose 8 to insert a word**

**Choose 9 to get length**

**Choose 10 to exit**

**Using  a do-while loop continue running this program until the user choose to exit.**

**Test your program on the following**

**Option     action**

**1**

**2         user enters Programming is great**

**4         replace    is   with     is not**

**3**

**6         append I love it a lot**

**3**

**5          look for the word not**

**7         delete not**

**9**

**8          insert after I  the word don’t**

**3**

**10**

 Code

MyMoto Class

**public** **class** MyMoto {

//data field moto of type string

**private** String moto;

// a default constructor to set data fields to spaces.

MyMoto(){

moto = " ";

}

//a method named getMoto() to display the moto

**public** String getMoto() {

**return** moto;

}

//a method named setMoto() to change the moto

**public** **void** setMoto(String newMoto) {

moto = newMoto;

}

//a method named replace to replace a word with another word in the moto

**public** **void** replace(String rep, String word) {

String temp = getMoto() ;

String t2 = temp.replaceAll(rep, word);

setMoto(t2);

}

// a method named matches to check if a word matches with a word in the moto

//return boolean value

**public** **boolean** matches(String w1) {

String w2 = ".\*"+ w1 + ".\*";

**return** moto.matches(w2);

}

// A method named append to append to the end of the moto

**public** **void** append(String word) {

String temp = getMoto();

setMoto(temp +" "+ word);

}

//A method named delete to remove a word from the moto

**public** **void** delete(String word) {

//find the word replace the word with spaces

String temp = getMoto() ;

String t2 = temp.replaceAll(word,"");

setMoto(t2);

}

//A method named insert to insert a word into the moto at a specific location

**public** **void** insert(String location, String word) {

//add %s to end of the locating wor

String test = location + " %s";

String temp = getMoto() ;

// use String replace to the newly created test at location

String t2 = temp.replaceAll(location,test);

//use the string format to add the word

String fin = String.*format*(t2, word );

setMoto(fin);

/\*

//find location of the word store temp index of the char

int loc =0;

char [] chars = getMoto().toCharArray();

int tempIndex = 0;

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

if (chars[i] == location.charAt(0)) {

tempIndex = i;

}

}

loc = tempIndex + 2;

//using String builder to insert the word to location

StringBuilder str = new StringBuilder (getMoto());

String w1 = word + " " ;

String st1 = str.insert(loc,w1).toString();

setMoto(st1);\*/

}

//a method named length to get the length of the moto

**public** **int** length() {

**return** moto.length();

}

}

Main Class

**import** java.util.Scanner;

**public** **class** Main\_MyMoto {

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

Scanner input1 = **new** Scanner (System.***in***);

MyMoto m1 = **new** MyMoto ();

**boolean** exit = **true**;

**do** {

**int** option = *menu* ();

**switch** (option) {

**case** 1 :

//MyMoto m1 = new MyMoto ();

**break**;

**case** 2 :

System.***out***.println("Enter to set the moto:");

m1.setMoto(input1.nextLine());

**break**;

**case** 3 :

System.***out***.println(m1.getMoto());

**break**;

**case** 4 :

System.***out***.println("Word that you want replace in your Moto: ");

String s1 = input1.nextLine();

System.***out***.println("Replacement word: ");

String s2 = input1.nextLine();

m1.replace(s1,s2);

**break**;

**case** 5:

System.***out***.println("Look up word");

String match = input1.next();

System.***out***.println(m1.matches(match) ? "Found ": "Not Found");

**break**;

**case** 6:

System.***out***.println("Enter a word/sentance to add to the end of your moto");

String sen = input1.nextLine();

m1.append(sen);

**break**;

**case** 7:

System.***out***.println("Enter a word to remove from your Moto: ");

String del = input1.next();

m1.delete(del);

**break**;

**case** 8:

System.***out***.println("Word you wish to insert: ");

String insert = input1.next();

System.***out***.println("Location that you wish to insert");

String location = input1.next();

m1.insert(location, insert);

**break**;

**case** 9:

System.***out***.println("Length for your Moto:"

+ m1.length());

**break**;

**case** 10:

exit = **false** ;

**break**;

}

}

**while** (exit);

}

**public** **static** **int** menu() {

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

System.***out***.println("Choose the Following option for My Moto");

System.***out***.println("Option 1 to create a new moto");

System.***out***.println("Option 2 to set moto");

System.***out***.println("Option 3 to get moto");

System.***out***.println("Option 4 to replace a word");

System.***out***.println("Option 5 to match a word");

System.***out***.println("Option 6 to append to moto");

System.***out***.println("Option 7 to delete a word");

System.***out***.println("Option 8 to insert a word at a Particular Location");

System.***out***.println("Option 9 to get the lenght ");

System.***out***.println("Option 10 to exit");

**int** choice = input.nextInt();

**return** choice;

}

}

Screenshot

