|  |
| --- |
| Staffordshire University |
| Further Object-Orientated Programming : CE00527-2-190909-161209 |
| Tutor: Cathy French |

|  |
| --- |
| Alex J Davison DV003874  1/11/2010 |

Contents

[Tutorial 6 2](#_Toc250904400)

[A description (with code snippets) of the changes you made to the PetShop class to use a ListIterator 2](#_Toc250904401)

[Code 2](#_Toc250904402)

[Test 2](#_Toc250904403)

[A class diagram for your class which represents a queue 2](#_Toc250904404)

[The code listing for your class which represents a queue, and you’re your answers to questions 3 and 4 2](#_Toc250904405)

[Question 3 2](#_Toc250904406)

[Question 4 2](#_Toc250904407)

[Tutorial 7 2](#_Toc250904408)

[The new methods that you added to your PetShop system to save and load pet information in Part 1 2](#_Toc250904409)

[Tutorial 8 2](#_Toc250904410)

[A description of the changes you have made, including a listing of any new code, to the PetShop system to read and write the list of Pets to file 2](#_Toc250904411)

[Code 2](#_Toc250904412)

[Test 2](#_Toc250904413)

[Code listing and output for all the exercises in Part 2 2](#_Toc250904414)

[Code 2](#_Toc250904415)

[Tutorial 9 2](#_Toc250904416)

[Code listing, output, and answers to any questions for all the exercises 2](#_Toc250904417)

[Question 1 2](#_Toc250904418)

[Question 2 2](#_Toc250904419)

[Question 3 2](#_Toc250904420)

[Tutorial 10 2](#_Toc250904421)

[Code listing, evidence of testing answers to questions for all exercises in Part 1 2](#_Toc250904422)

[Question 1 2](#_Toc250904423)

[Question 2 2](#_Toc250904424)

[Question 3 2](#_Toc250904425)

[Question 4 2](#_Toc250904426)

[The code listing for any new classes you created to implement the game in Part 2, plus any code you added to the existing classes 2](#_Toc250904427)

[Evidence (including screen shots) of testing your game 2](#_Toc250904428)

[Tutorial 11 2](#_Toc250904429)

[Code listing and evidence of testing for all exercises 2](#_Toc250904430)

[Question 1 2](#_Toc250904431)

[Question 2/3/4 2](#_Toc250904432)

# Tutorial 6

## A description (with code snippets) of the changes you made to the PetShop class to use a ListIterator

### Code

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

import java.util.\*;

import java.io.\*;

import java.io.Serializable;

/\*\*

\*

\* @author Alex J Davison

\*/

public class PetShop implements Serializable {

private ArrayList<Pet> thePets;

private ListIterator <Pet> theIterator;

private String thePetShopName;

private Pet currentPet;

public PetShop(String name) {

// call the read from text method

thePetShopName = name;

thePets = new ArrayList<Pet>();

theIterator = thePets.listIterator();

}

public Pet getCurrentPet() {

return currentPet;

}

public void nextPet() {

if (theIterator.hasNext())

currentPet = theIterator.next();

}

public void previousPet() {

if (theIterator.hasPrevious())

currentPet = theIterator.previous();

}

public boolean addPet(Pet newPet) {

theIterator.add(newPet);

return true;

}

public void outputPetShopPets() {

System.out.println("Pet Shop Name: " + thePetShopName);

//output each player in turn

System.out.println("The pets:");

for (Pet nextPet : thePets) {

System.out.println(nextPet.toString());

}

}

public Pet findPet(String petName) {

for (Pet p : thePets) {

if (p.getName().equalsIgnoreCase(petName)) {

currentPet = thePets.get(thePets.indexOf(p));

return p;

}

}

return null;

}

public boolean deletePet(String name) {

Pet p = findPet(name);

if (p != null) {

theIterator.remove();

return true;

}

return false;

}

}

### Test

public class TestIterator {

public static void main(String[] args) {

PetShop thePetShop = new PetShop("Stafford Octagons");

Cat c = new Cat("Alex", false, "01/29/1987",true);

Dog d = new Dog("Dainie", false, "07/07/1990","Lab");

Hamster h = new Hamster("Ed", false, "07/07/1990",1.5);

Parrot p = new Parrot("Ed", false, "07/07/1990","Yellow");

thePetShop.addPet(c);

thePetShop.addPet(d);

thePetShop.addPet(h);

thePetShop.addPet(p);

thePetShop.outputPetShopPets();

thePetShop.previousPet();

System.out.println(thePetShop.getCurrentPet());

thePetShop.previousPet();

System.out.println(thePetShop.getCurrentPet());

thePetShop.previousPet();

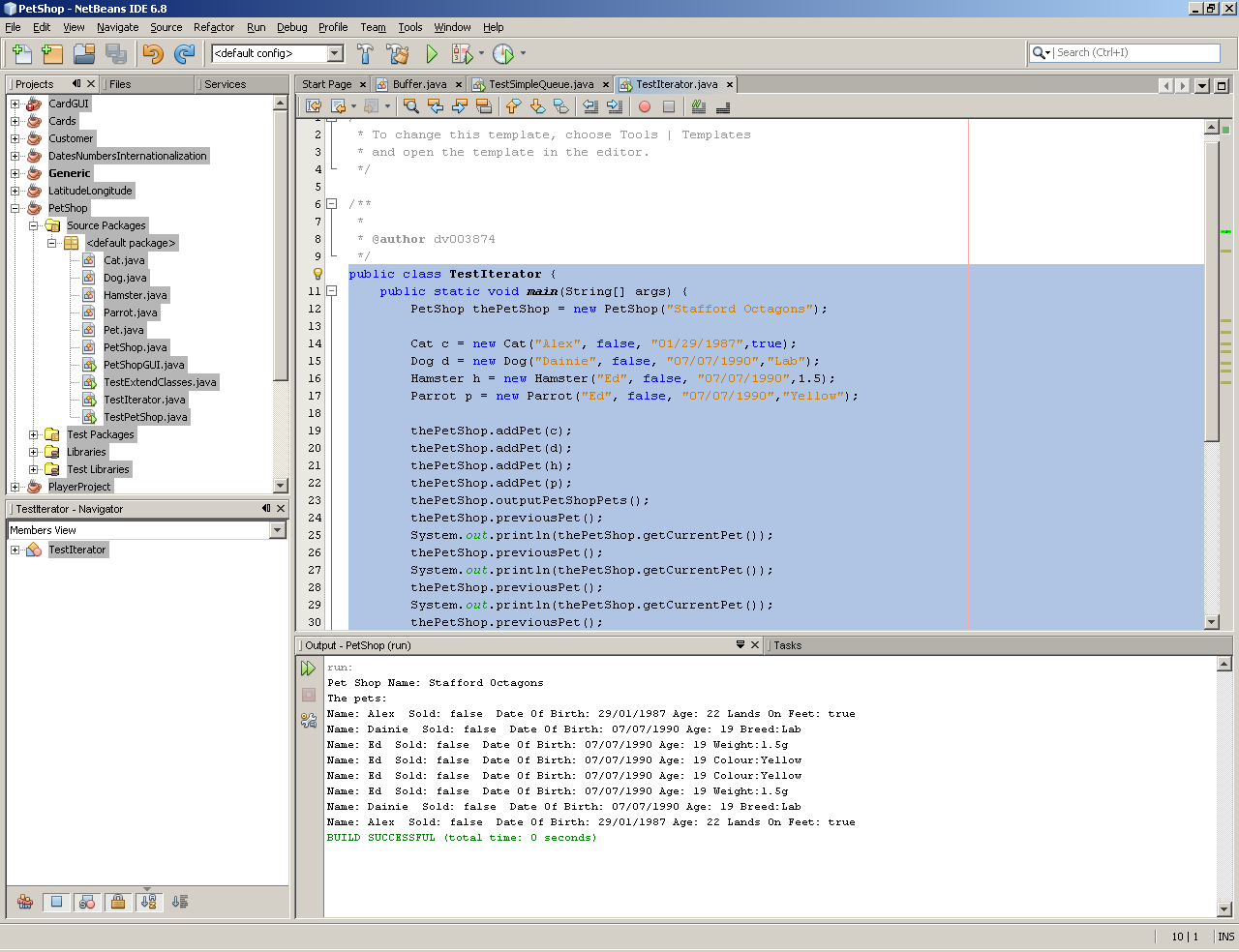
System.out.println(thePetShop.getCurrentPet());

thePetShop.previousPet();

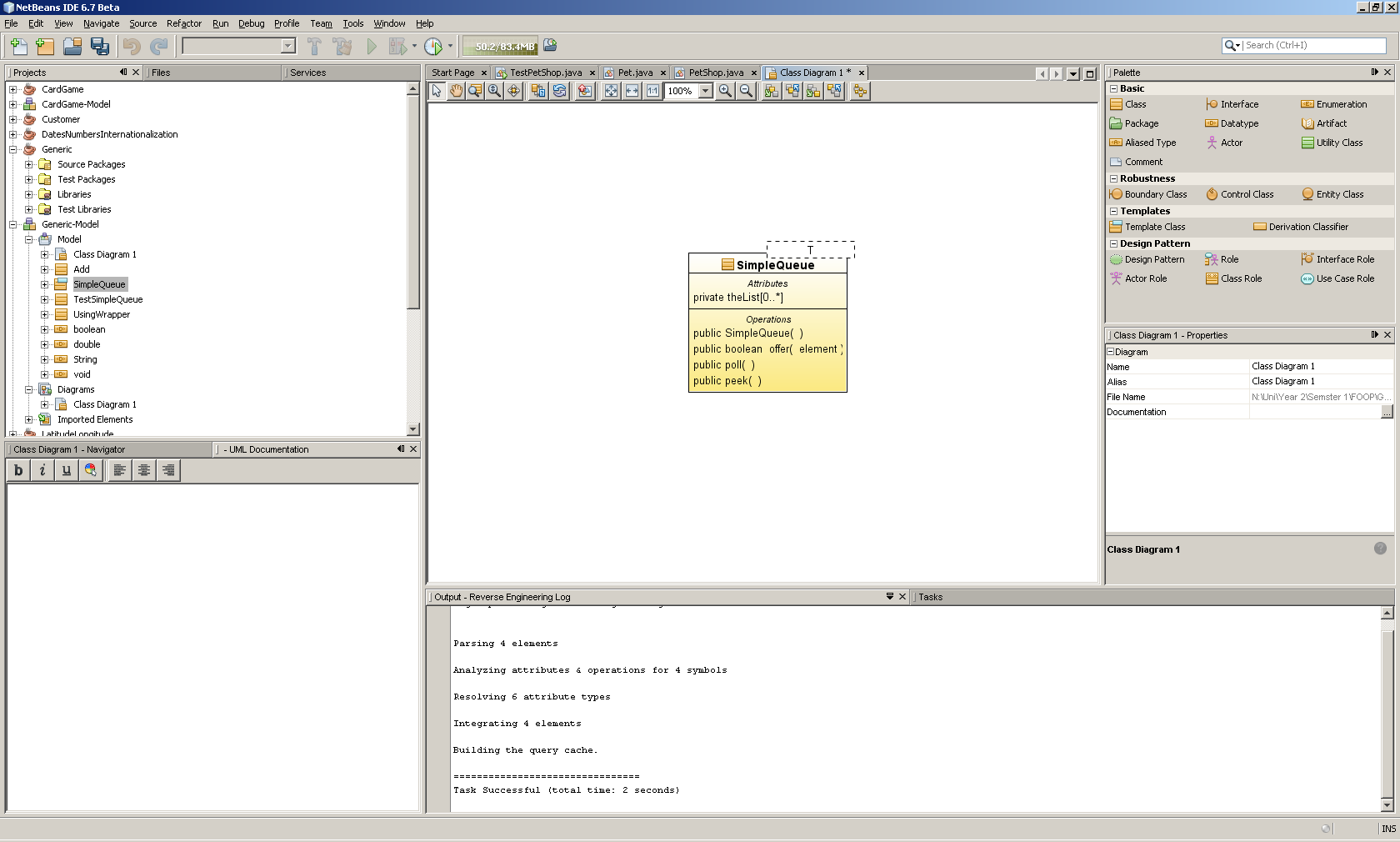
System.out.println(thePetShop.getCurrentPet());

}

}



## A class diagram for your class which represents a queue



## The code listing for your class which represents a queue, and you’re your answers to questions 3 and 4

### Question 3

#### Code

public class UsingWrapper {

public static void main(String[] args) {

ArrayList<Double> theList = new ArrayList<Double>();

Scanner kybd = new Scanner(System.in);

for (int i = 0; i < 10; i++){

System.out.print("Enter a double number: ");

theList.add(kybd.nextDouble());

}

System.out.println("Numbers in reverse");

for (int i = 9; i >= 0; i--){

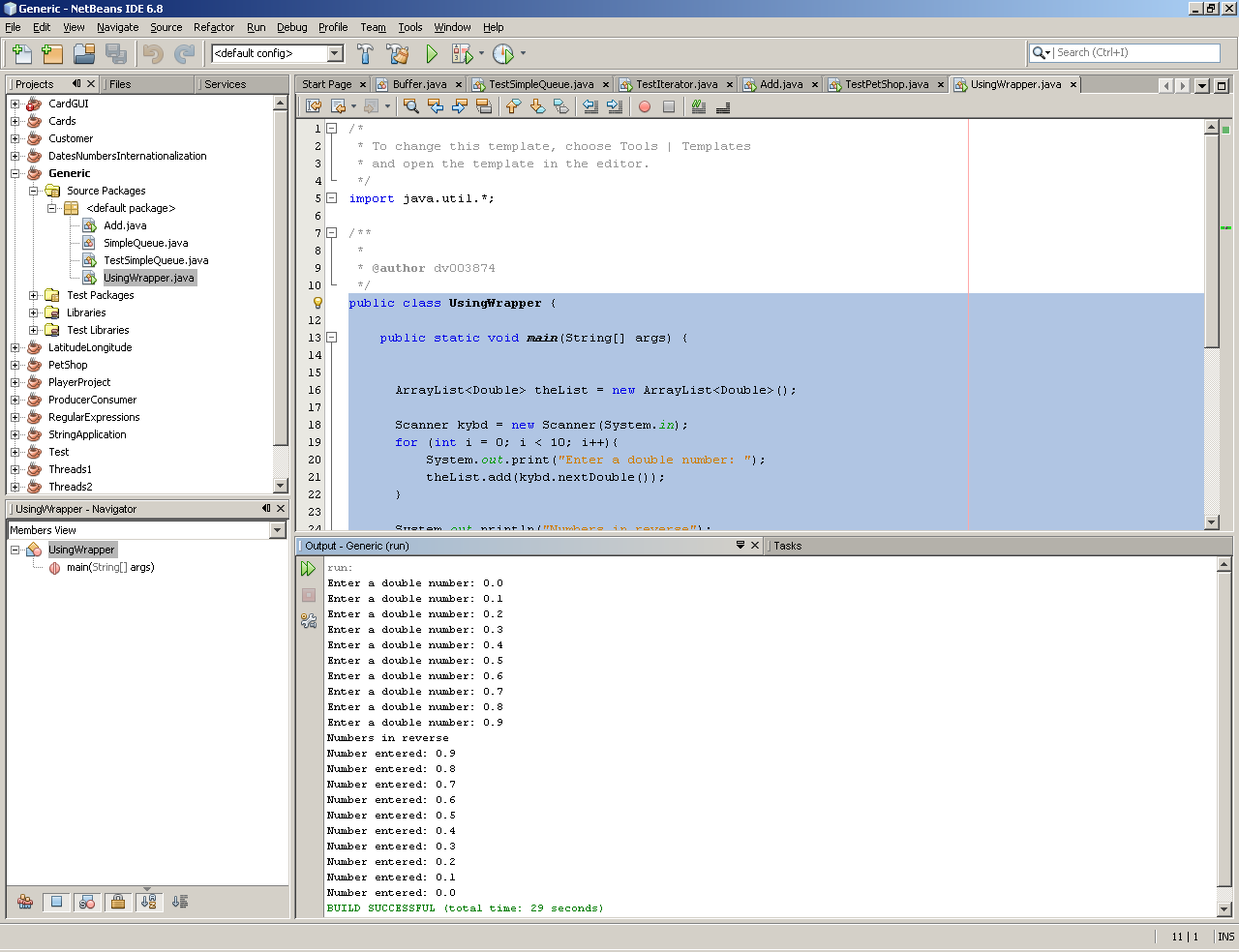
System.out.println("Number entered: " +theList.get(i));

}

}

}

#### Test



### Question 4

#### Code

public class Add {

public static void main(String[] args) {

Double answer1 = add(4, 3);

Number answer2 = add(new Integer(9), new Double(7.2));

double answer3 = add(5.1, 13);

System.out.println(answer1 + " " + answer2 + " " + answer3);

}

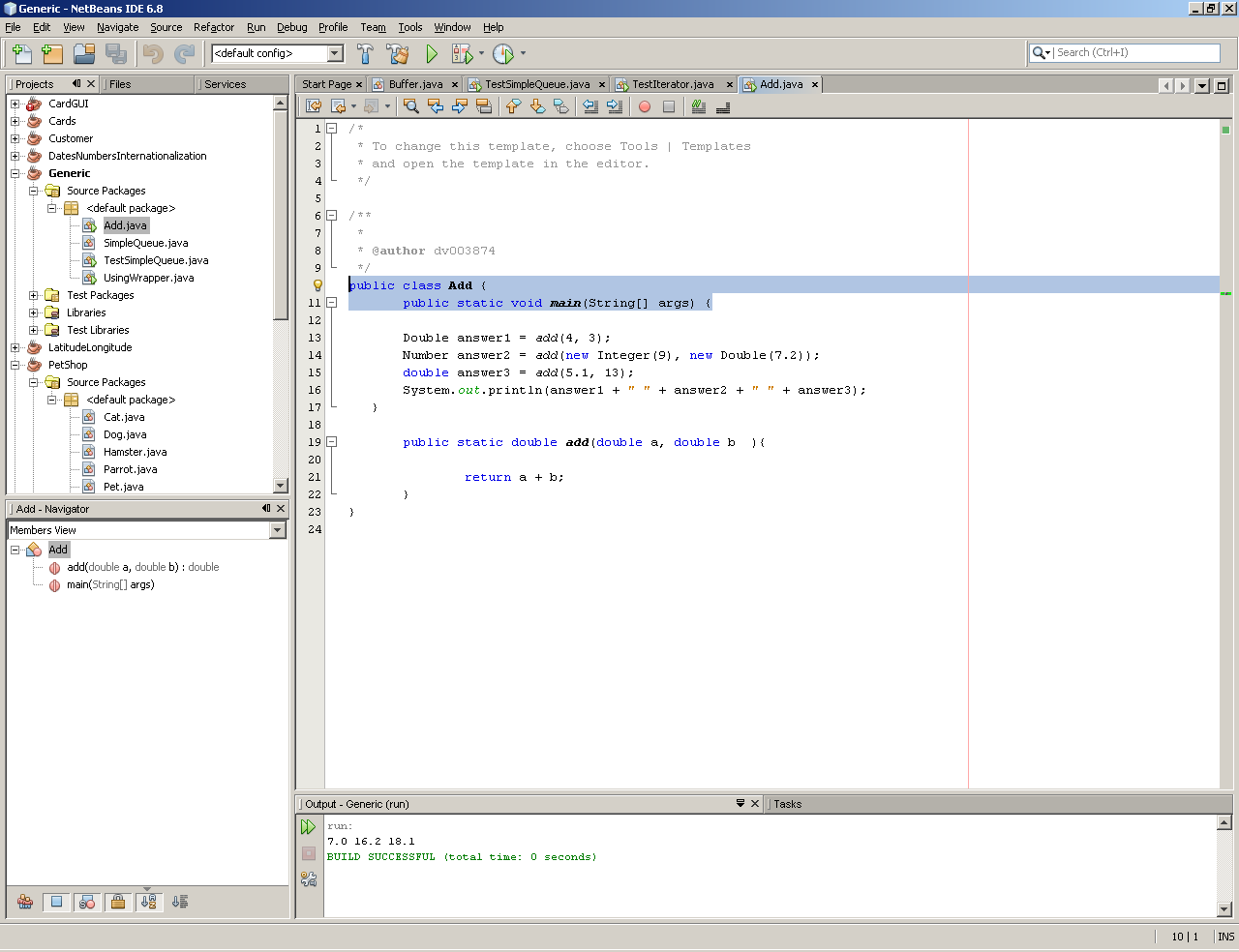
public static double add(double a, double b ){

return a + b;

}

}

#### Test



# Tutorial 7

## The new methods that you added to your PetShop system to save and load pet information in Part 1

public void saveCurrentPetText(){

if(currentPet !=null){

try {

File currentPetFile = new File("currentPetFile.txt");

FileWriter fw = new FileWriter(currentPetFile,true);

currentPet.savePetText(fw);

fw.flush(); // flush before closing

fw.close(); // close file when done

}catch(IOException e) { }

}

}

public String addTextFile (String filename){

Scanner input = null;

String type = null;

String name = null;

Boolean sold = null;

String dob = null;

try {

input = new Scanner(new File(filename));

// input.useDelimiter(",");

while (input.hasNext()){

type = input.next();

if(type.equalsIgnoreCase("Cat"))

{

name = input.next();

sold = input.nextBoolean();

dob = input.next();

Boolean feet = input.nextBoolean();

Cat c1 = new Cat(name,sold,dob,feet);

this.addPet(c1);

}else if(type.equalsIgnoreCase("Dog"))

{

name = input.next();

sold = input.nextBoolean();

dob = input.next();

String breed= input.next();

Dog d1 = new Dog(name,sold,dob,breed);

this.addPet(d1);

}else if(type.equalsIgnoreCase("Hamster"))

{

name = input.next();

sold = input.nextBoolean();

dob = input.next();

double weight = input.nextDouble();

Hamster h1 = new Hamster(name,sold,dob,weight);

this.addPet(h1);

}else if(type.equalsIgnoreCase("Parrot"))

{

name = input.next();

sold = input.nextBoolean();

dob = input.next();

String colour = input.next();

Parrot p1 = new Parrot(name,sold,dob,colour);

this.addPet(p1);

}

else

{

System.out.println("Don't know what it is");

}

}

return "Pet(s) from file "+ filename +" has been added to the petshop.";

}

catch (FileNotFoundException ex) {

return "File does not exist: " + ex.getMessage();}

catch (Exception ex) {

return "Error reading data " + ex.toString();}

finally {

if (input != null)

input.close();

}

}

# Tutorial 8

## A description of the changes you have made, including a listing of any new code, to the PetShop system to read and write the list of Pets to file

### Code

public void serialOutFile() {

try{

DataOutputStream output = new DataOutputStream(new FileOutputStream("temp.dat"));

for (Pet nextPet : thePets) {

nextPet.savePetSerial(output);

}

output.close();

}catch(IOException e) { }

}

public void serialInFile(){

boolean EOF = false;

String type = null;

String name = null;

Boolean sold = null;

String dob = null;

try{

DataInputStream input = new DataInputStream(new FileInputStream("temp.dat"));

while (!EOF)

{

type = input.readUTF();

System.out.println(type);

if(type.equalsIgnoreCase("Cat"))

{

name = input.readUTF();

sold = input.readBoolean();

dob = input.readUTF();

Boolean feet = input.readBoolean();

Cat c1 = new Cat(name,sold,dob,feet);

this.addPet(c1);

}else if(type.equalsIgnoreCase("Dog"))

{

name = input.readUTF();

sold = input.readBoolean();

dob = input.readUTF();

String breed= input.readUTF();

Dog d1 = new Dog(name,sold,dob,breed);

this.addPet(d1);

}else if(type.equalsIgnoreCase("Hamster"))

{

name = input.readUTF();

sold = input.readBoolean();

dob = input.readUTF();

double weight = input.readDouble();

Hamster h1 = new Hamster(name,sold,dob,weight);

this.addPet(h1);

}else if(type.equalsIgnoreCase("Parrot"))

{

name = input.readUTF();

sold = input.readBoolean();

dob = input.readUTF();;

String colour = input.readUTF();

Parrot p1 = new Parrot(name,sold,dob,colour);

this.addPet(p1);

}

else

{

System.out.println("Don't know what it is");

}

}

}catch(IOException e) {

EOF = true;

}

}

### Test

public class TestPetShop {

public static void main(String[] args) {

PetShop thePetShop = new PetShop("Stafford Octagons");

Cat c1 = new Cat("Dave",false, "12/01/1990",true);

Dog d1 = new Dog("Murphy",true, "08/12/2006","Lab");

Hamster h1 = new Hamster("Fluffy",false,"09/25/2009",29.0);

Parrot p1 = new Parrot("Sam",false,"12/12/2006","Red");

thePetShop.addPet(c1);

thePetShop.addPet(d1);

thePetShop.addPet(h1);

thePetShop.addPet(p1);

thePetShop.outputPetShopPets();

System.out.println(thePetShop.addTextFile("currentPetFile.txt"));

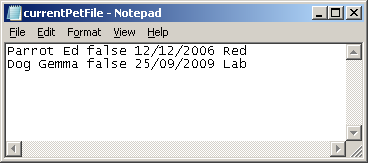
thePetShop.outputPetShopPets();

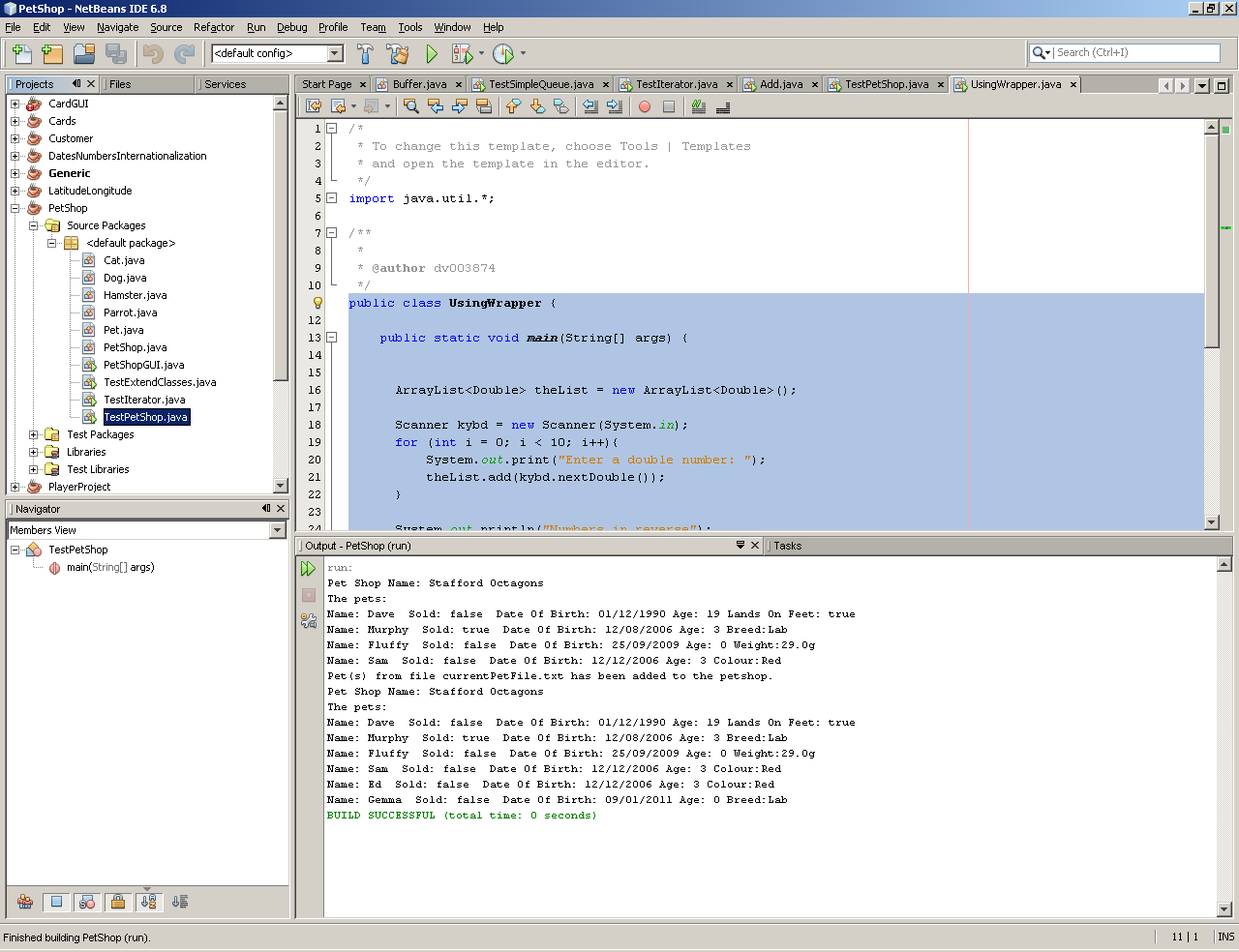
thePetShop.previousPet();

thePetShop.saveCurrentPetText();

}

}





## Code listing and output for all the exercises in Part 2

### Code

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

StringBuilder s1 = new StringBuilder("dislocated");

System.out.println(s1);

s1.delete(0, 3);

s1.delete(5, 7);

s1.insert(0,"auto");

s1.insert(s1.length(),"ion");

System.out.println(s1);

StringBuilder s2 = new StringBuilder("margarine");

System.out.println(s2);

s2.delete(2,3);

s2.delete(4,5);

s2.insert(4,"z");

System.out.println(s2);

StringBuilder s3 = new StringBuilder("alex davison love's dainie smith");

System.out.println(s3);

s3.delete(0,12);

s3.insert(0, "murphy");

System.out.println(s3);

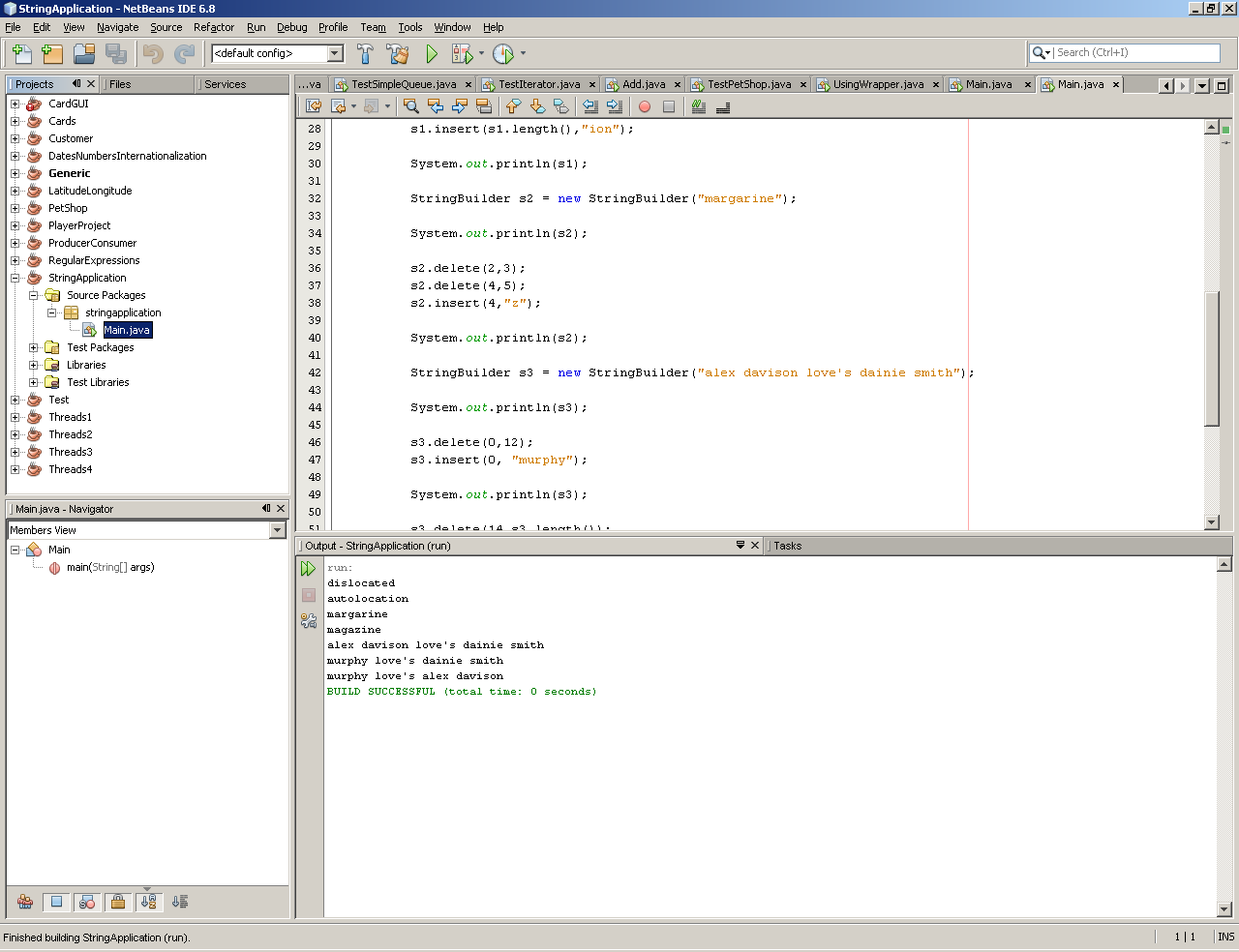
s3.delete(14,s3.length());

s3.insert(14, "alex davison");

System.out.println(s3);

}

}



# Tutorial 9

## Code listing, output, and answers to any questions for all the exercises

### Question 1

#### Code

import java.util.Scanner;

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

String myName = "(?i).\*alex.\*";

String strInput;

Scanner input = new Scanner(System.in);

System.out.println("Please enter your full name:");

strInput = input.nextLine();

System.out.println("The input has my name in it " + strInput.matches(myName));

while (strInput.compareTo("exit")!=0){

System.out.println("Please enter your full name:");

strInput = input.nextLine();

System.out.println("The input has my name in it " + strInput.matches(myName));

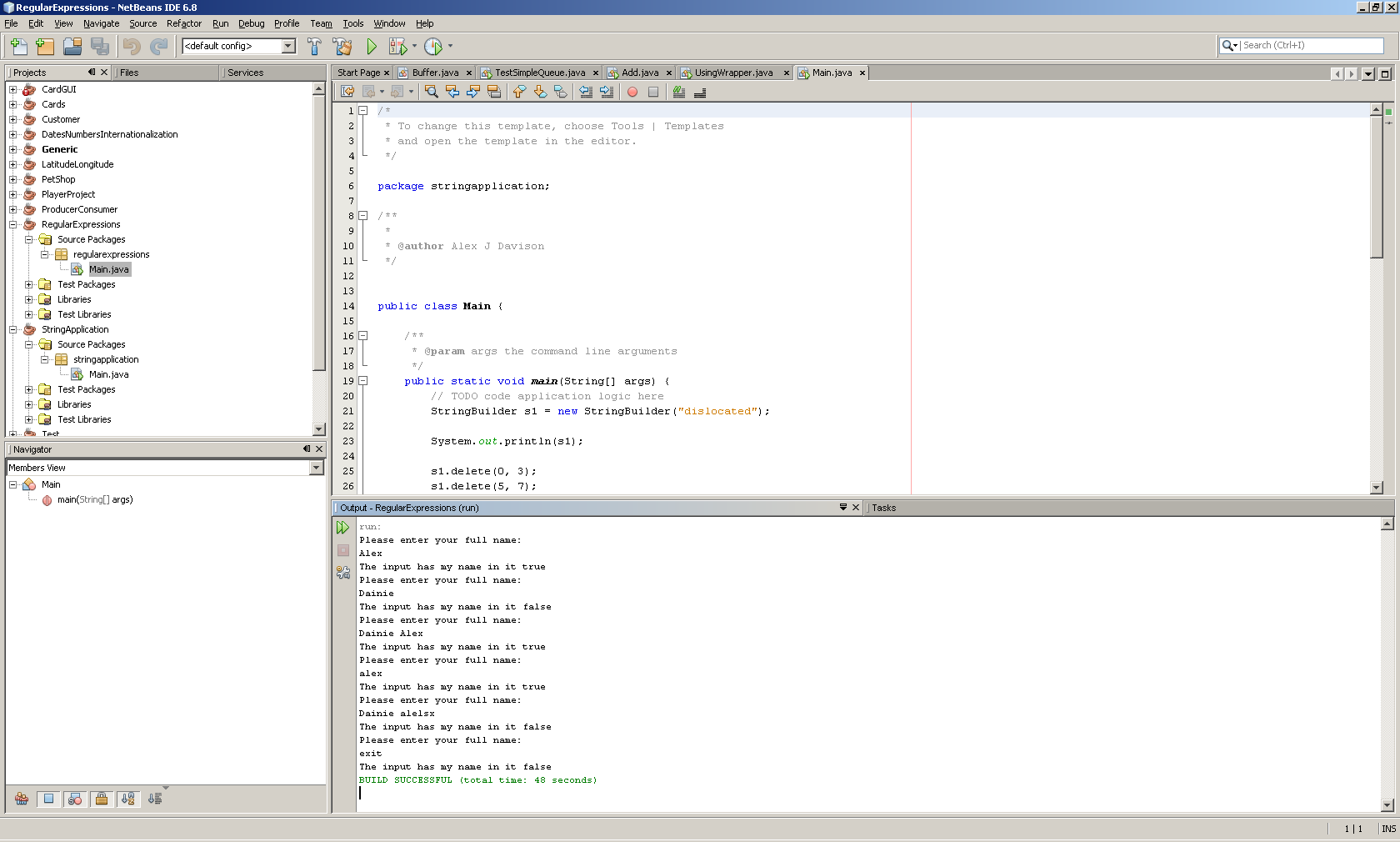
}

input.close();

}

}

#### Test



### Question 2

#### Code

import java.util.\*;

import java.util.regex.\*;

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Pattern p = Pattern.compile("\\d+\\W{1}\\d+;[-]?\\d+\\W{1}\\d+");

String s = "52.436146666666666;-2.2541 6 @ %52.43593333333333;-2.2543816666666667 7 H 0 52.43570166666667;-2.254615 8 ˆ @ %52.43544166666667;-2.2548066666666666 9 ¸ @ &52.435188333333336;-2.2546233333333334 : ø 0 52.43495;-2.25441 ; 8 0 52.43469;-2.254261666666667 < h 0 52.43442;-2.25422";

Matcher m = p.matcher(s);

Scanner scan;

while (m.find())

{

scan=new Scanner(m.group());

scan.useDelimiter(";");

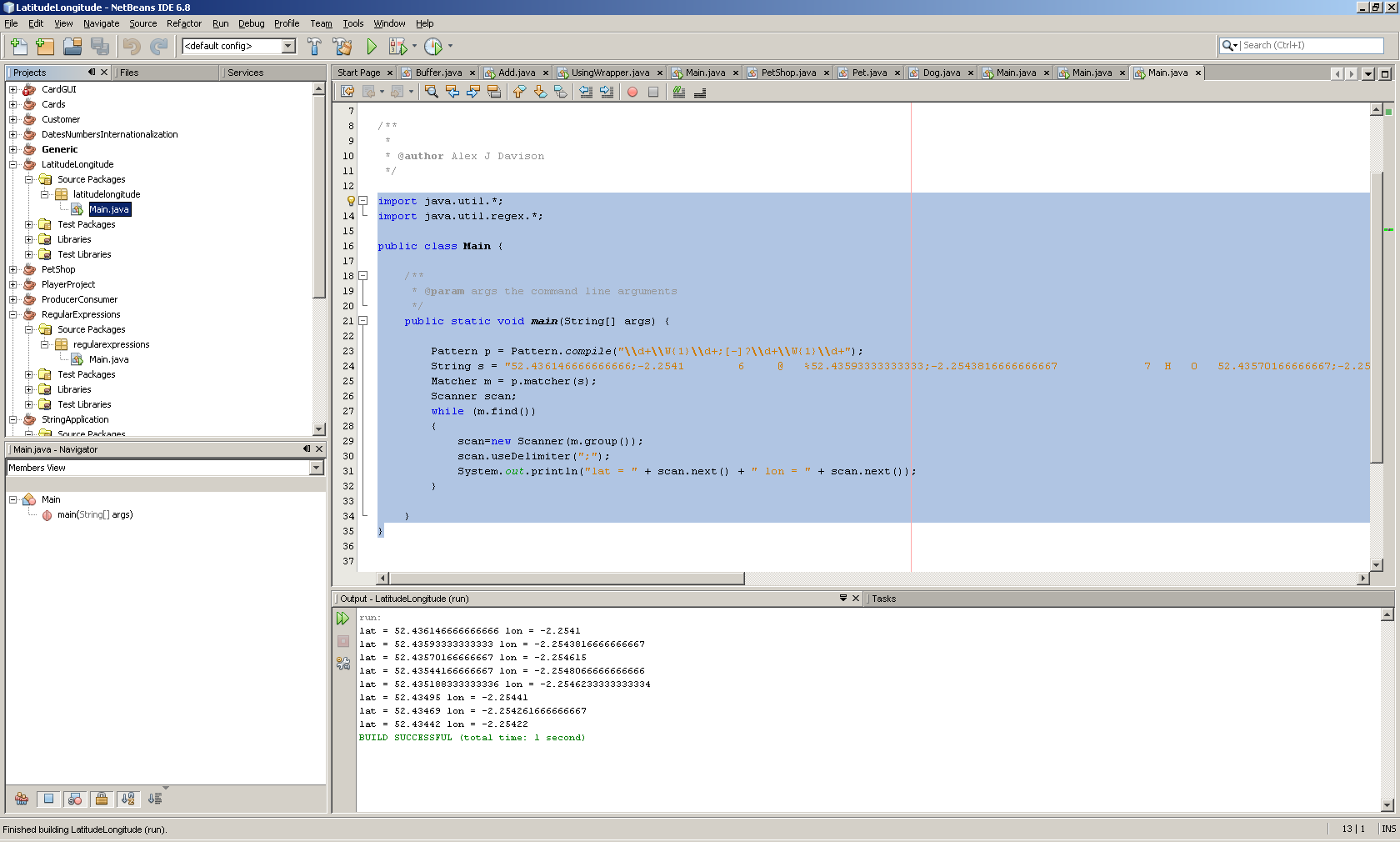
System.out.println("lat = " + scan.next() + " lon = " + scan.next());

}

}

}

#### Test



### Question 3

#### Code

import java.util.\*;

import java.text.DateFormat;

import java.text.NumberFormat;

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

DateFormat shortDate = DateFormat.getDateInstance(DateFormat.SHORT);

DateFormat longDate = DateFormat.getDateInstance(DateFormat.LONG);

Date d1 = new Date(); // todays date

Calendar c1 = Calendar.getInstance();

c1.setTime(d1);

c1.add(Calendar.YEAR,2); // add 6 months

Date d2 = c1.getTime();

System.out.println("Short");

System.out.println("Now it is " + shortDate.format(d1));

System.out.println("In 2 years it will be " + shortDate.format(d2));

System.out.println("Long");

System.out.println("Now it is " + longDate.format(d1));

System.out.println("In 2 years it will be " + longDate.format(d2));

Locale locale[];

locale = Calendar.getAvailableLocales();

for (Locale loc: locale)

{

DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.SHORT, loc);

NumberFormat perFormat = NumberFormat.getPercentInstance(loc);

NumberFormat numFormat = NumberFormat.getNumberInstance(loc);

NumberFormat curFormat = NumberFormat.getCurrencyInstance(loc);

System.out.println(loc.getDisplayName() +

"\nDate: " + dateFormat.format(d1)+

"\nPercentage: " + perFormat.format(23.45) +

"\nNumber: " + numFormat.format(102.23) +

"\nCurreny: " + curFormat.format(5142.465));

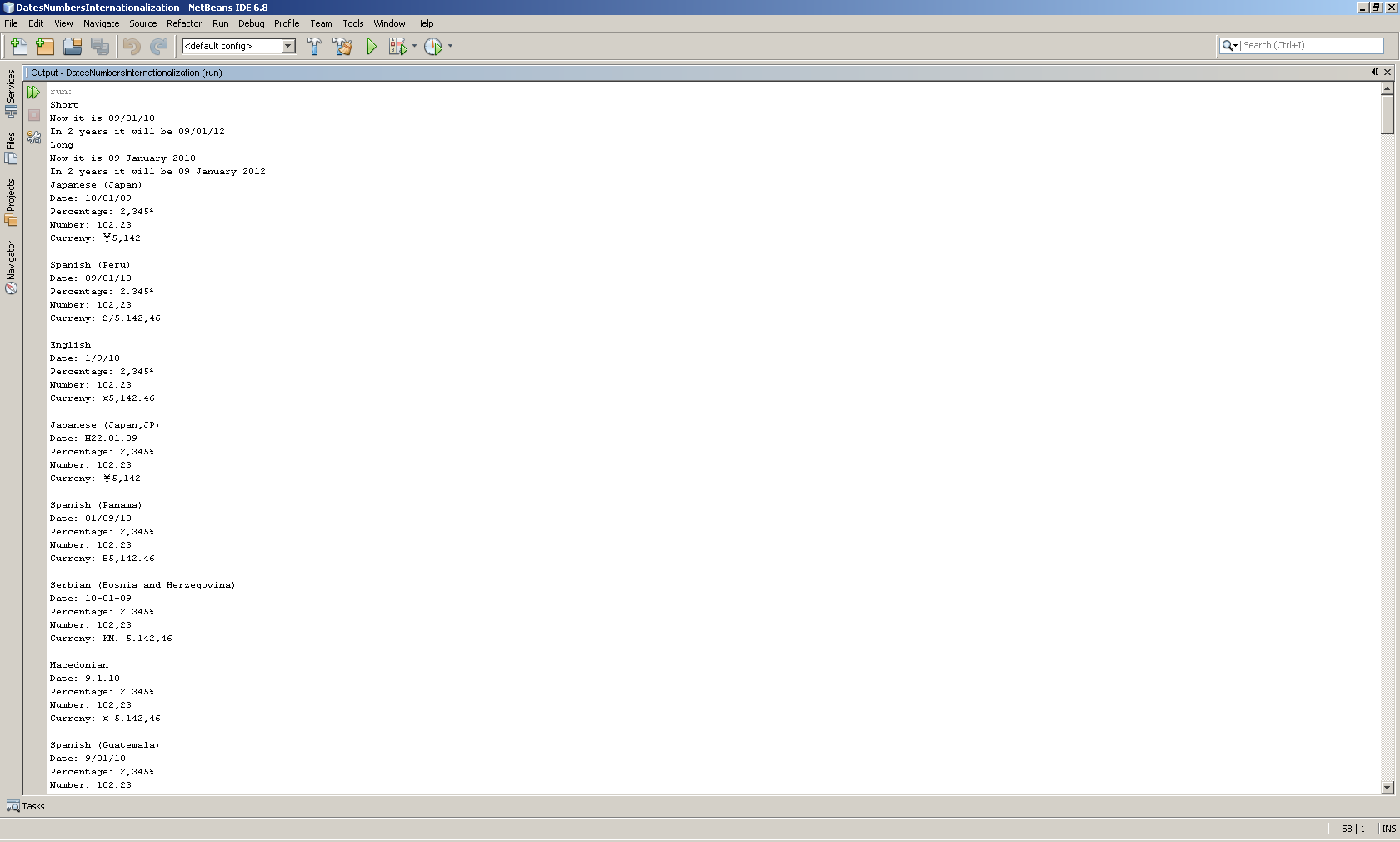
System.out.println();

}

}

}

#### Test



# Tutorial 10

## Code listing, evidence of testing answers to questions for all exercises in Part 1

### Question 1

#### Code

public class MyThread {

private String msg;

private int num;

public MyThread(String m, int n)

{

msg = m;

num = n;

}

public void run(){

// the work of the thread takes place within the run method

// the thread terminates when the method ends

for (int i = 0; i < num; ++i)

{

System.out.println(msg + " Count: " + (i+1) + "/" + num);

}

}

}

#### Test

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

int numThreads = 5;

MyThread threads[];

String msg[];

msg = new String [numThreads];

threads = new MyThread[numThreads];

for (int i = 0; i < numThreads; i++)

{

msg[i] = "Thread: " + i;

threads[i] = new MyThread(msg[i],numThreads \* i + 1);

}

for (int i = 0; i < numThreads; i++)

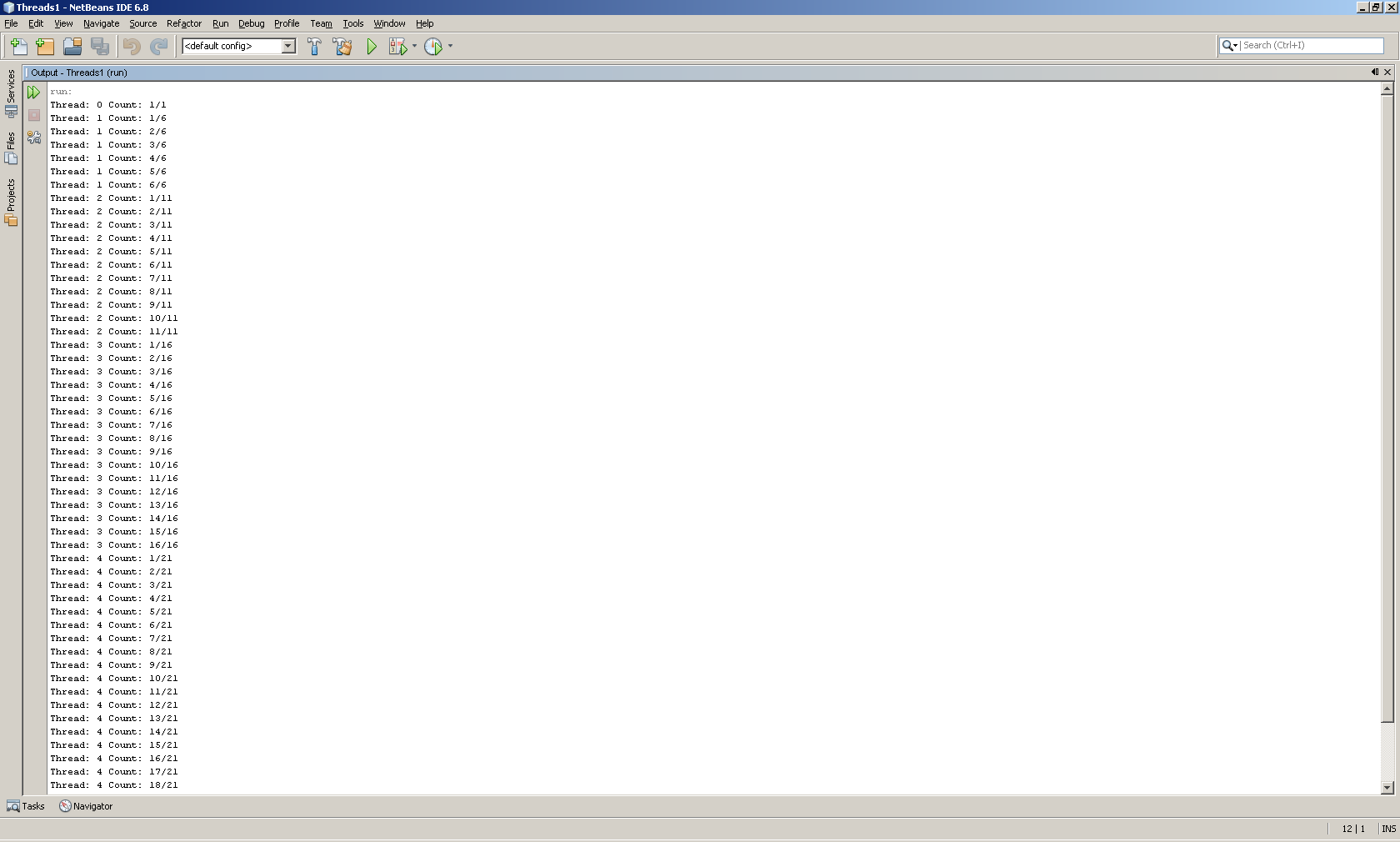
{

threads[i].run();

}

}

}



### Question 2

#### Code

public class MyThread extends Thread {

private String msg;

private int num;

public MyThread(String m, int n)

{

msg = m;

num = n;

}

public void run(){

// the work of the thread takes place within the run method

// the thread terminates when the method ends

for (int i = 0; i < num; ++i)

{

System.out.println(msg + " Count: " + (i+1) + "/" + num) ;

}

}

}

#### Test

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

int numThreads = 5;

MyThread threads[];

String msg[];

msg = new String [numThreads];

threads = new MyThread[numThreads];

for (int i = 0; i < numThreads; i++)

{

msg[i] = "Thread: " + i;

threads[i] = new MyThread(msg[i],numThreads \* i + 1);

}

for (int i = 0; i < numThreads; i++)

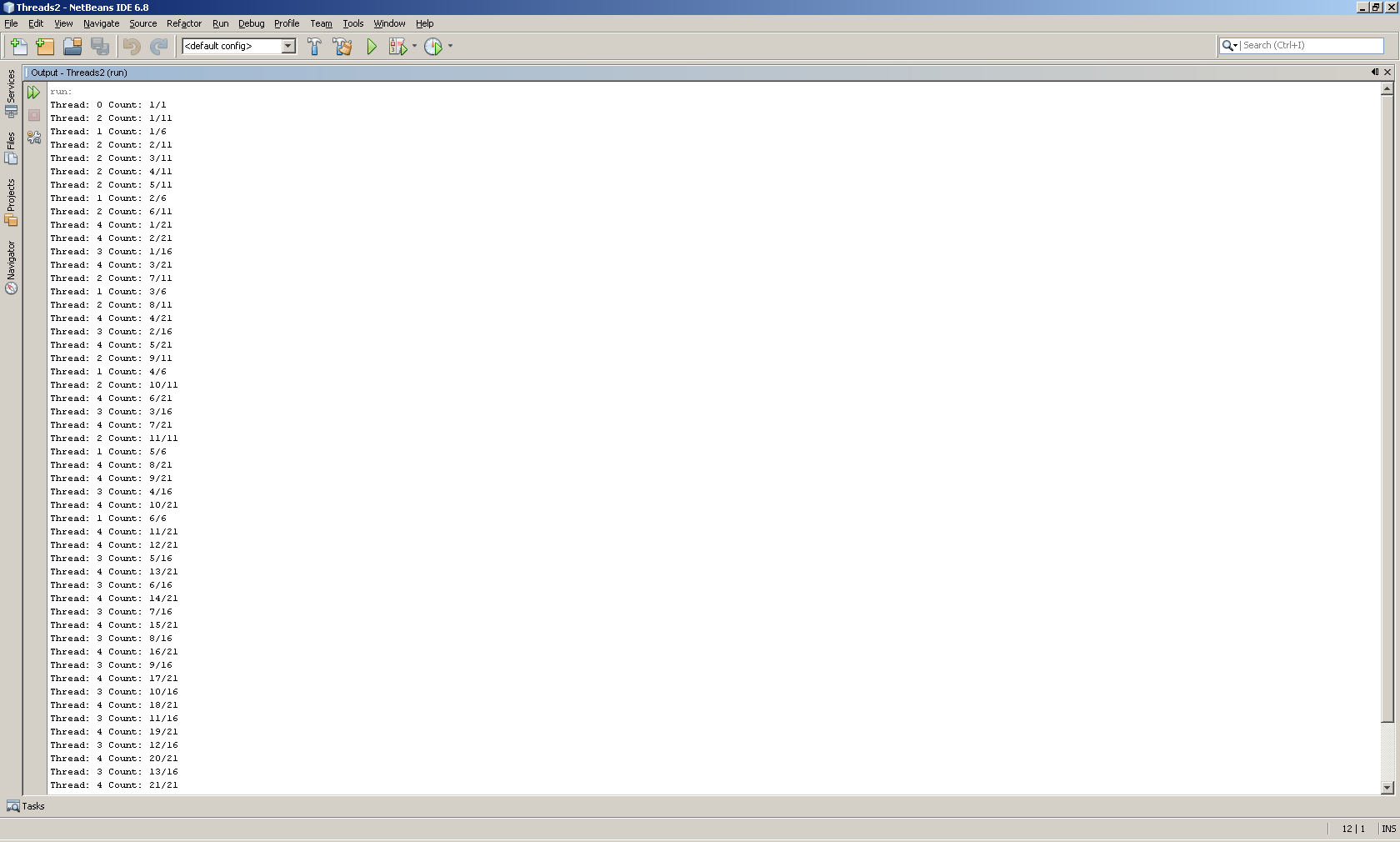
{

threads[i].start();

}

}

}



### Question 3

#### Code

public class MyThread extends Thread {

private String msg;

private int num;

public MyThread(String m, int n)

{

msg = m;

num = n;

}

public void run(){

// the work of the thread takes place within the run method

// the thread terminates when the method ends

for (int i = 0; i < num; ++i)

{

System.out.println(msg + " Count: " + (i+1) + "/" + num) ;

try

{

Thread.sleep(100);

}

catch (InterruptedException e)

{

System.out.println("Thread interrupted: " + e);

}

}

}

}

#### Test

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

int numThreads = 5;

MyThread threads[];

String msg[];

msg = new String [numThreads];

threads = new MyThread[numThreads];

for (int i = 0; i < numThreads; i++)

{

msg[i] = "Thread: " + i;

threads[i] = new MyThread(msg[i],numThreads \* i + 1);

}

for (int i = 0; i < numThreads; i++)

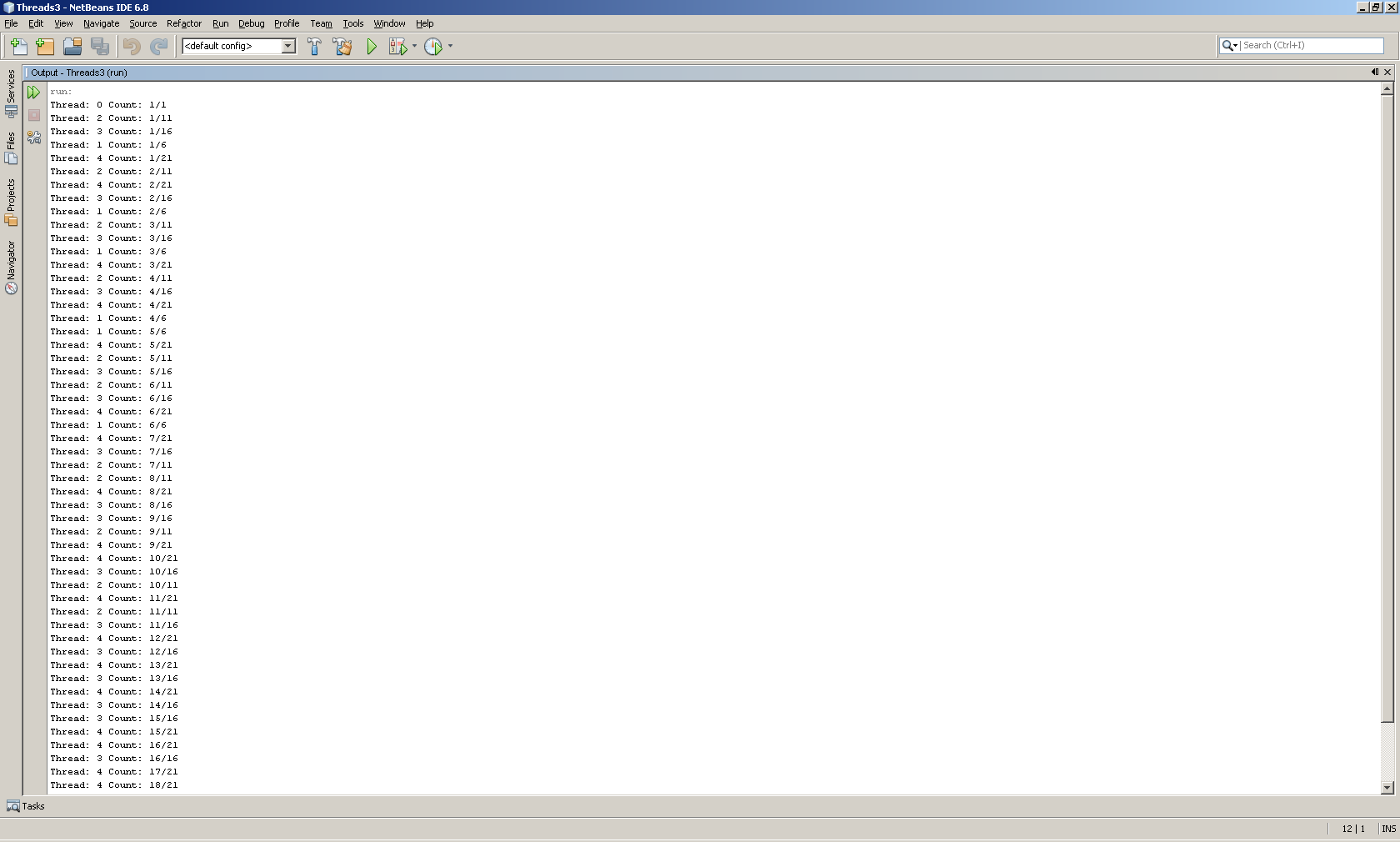
{

threads[i].start();

}

}

}



### Question 4

#### Code

public class MyThread extends Thread {

private String msg;

private int num;

public MyThread(String m, int n)

{

msg = m;

num = n;

}

public void run(){

// the work of the thread takes place within the run method

// the thread terminates when the method ends

Random randomGenerator = new Random();

for (int i = 0; i < num; ++i)

{

System.out.println(msg + " Count: " + (i+1) + "/" + num) ;

try

{

int randomInt = randomGenerator.nextInt(1000);

Thread.sleep(randomInt);

}

catch (InterruptedException e)

{

System.out.println("Thread interrupted: " + e);

}

}

}

}

#### Test

public class MyThread extends Thread {

private String msg;

private int num;

public MyThread(String m, int n)

{

msg = m;

num = n;

}

public void run(){

// the work of the thread takes place within the run method

// the thread terminates when the method ends

Random randomGenerator = new Random();

for (int i = 0; i < num; ++i)

{

System.out.println(msg + " Count: " + (i+1) + "/" + num) ;

try

{

int randomInt = randomGenerator.nextInt(1000);

Thread.sleep(randomInt);

}

catch (InterruptedException e)

{

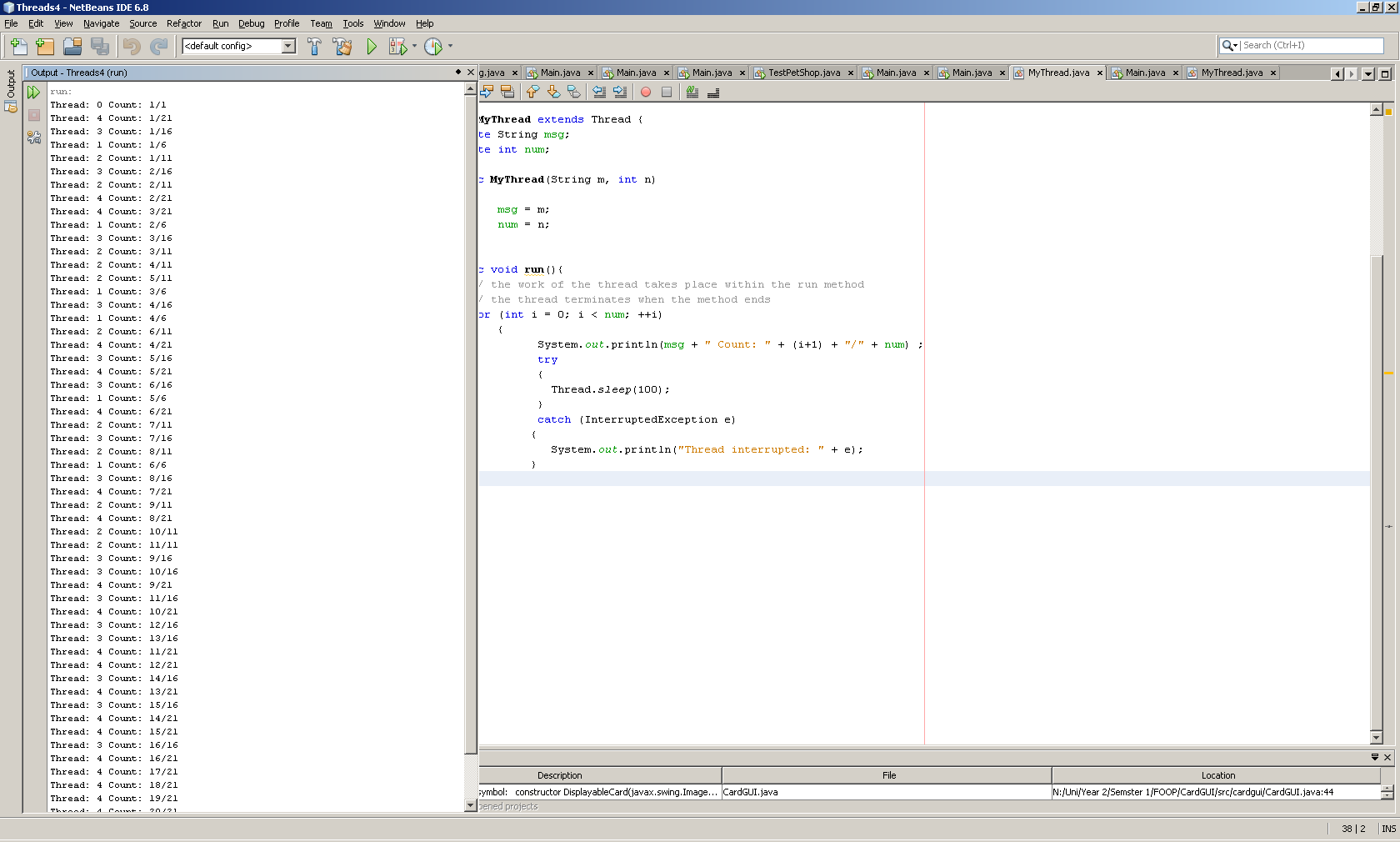
System.out.println("Thread interrupted: " + e);

}

}

}

}



## The code listing for any new classes you created to implement the game in Part 2, plus any code you added to the existing classes

import java.util.\*;

public class Pontoon {

public static void main(String[]args)

{

Scanner kybd = new Scanner(System.in);

String ans;

do

{

// play a hand of game

// ask user if finished

playHand();

System.out.println("Have you finished playing?");

ans = kybd.next();

} while ( ans.equals("no") );

}

public static void playHand()

{

System.out.println("Starting a new hand of the game");

Deck deck = new Deck();

deck.shuffle();

Hand player = new Hand();

Hand computer = new Hand();

player.addCard(deck.getCard(1));

computer.addCard(deck.getCard(1));

player.addCard(deck.getCard(1));

computer.addCard(deck.getCard(1));

System.out.println("Player's hand: ");

System.out.println(player.getCurrentHand());

System.out.println(player.getCurrentScore());

System.out.println("Computer's hand: ");

System.out.println(computer.getCurrentHand());

System.out.println(computer.getCurrentScore());

Scanner kybd = new Scanner(System.in);

if ( computer.getCurrentScore() > 21 )

{

System.out.println("Sorry computer is bust!");

// check and display the winner

checkWinner(computer.getCurrentScore(),player.getCurrentScore());

}

else if ( player.getCurrentScore() > 21 )

{

System.out.println("Sorry your bust!");

// check and display the winner

checkWinner(computer.getCurrentScore(),player.getCurrentScore());

}

else

{

// ask if card required

System.out.println("Would you like another card?");

String reply = kybd.next();

while ( reply.equals("yes"))

{

player.addCard(deck.getCard(1));

System.out.println("Player's hand: ");

System.out.println(player.getCurrentHand());

System.out.println(player.getCurrentScore());

if ( player.getCurrentScore() > 21 )

{

checkWinner(computer.getCurrentScore(),player.getCurrentScore());

reply = "no";

}

else

{

System.out.println("Would you like another card?");

reply = kybd.next();

}

}

if (player.getCurrentScore() <= 21 )

{

while ( computer.getCurrentScore() < 14 )

{

// deal a card to the computer and generate players score

System.out.print("Comptuer has another card.");

computer.addCard(deck.getCard(1));

System.out.println("Computer's hand: ");

System.out.println(computer.getCurrentHand());

System.out.println(computer.getCurrentScore());

}

}

checkWinner(computer.getCurrentScore(),player.getCurrentScore());

}

}

public static void checkWinner(int computerScore, int playerScore)

{

if (computerScore > 21 && playerScore > 21)

{

System.out.println("Computer/Player both lose.");

}

else if (computerScore > 21)

{

System.out.println("Congratulations player wins");

}

else if (playerScore > 21)

{

System.out.println("Sorry player lose, computer wins");

}

else if(playerScore == computerScore)

{

System.out.println("We player and computer wins!!!");

}

else if (playerScore > computerScore)

{

System.out.println("Congratulations player wins");

}

else

{

System.out.println("Sorry player lose, computer win");

}

}

}

public class Hand implements Cloneable{

// a class to represent a hand of cards

private ArrayList<Card> theCards;

public Hand() {

theCards = new ArrayList<Card>();

}

public void addCard(Card aCard) {

theCards.add(aCard);

}

public String toString() {

String s = "";

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

s += "\n" + theCards.get(i);

}

return s;

}

public String getCurrentHand() {

String s = "";

for (Card theCard : theCards) {

s = s + theCard;

}

return s;

}

public int getCurrentScore() {

int s = 0;

for (Card theCard : theCards) {

s = s + theCard.getRank().getNumScore();

}

return s;

}

public Object clone() throws CloneNotSupportedException

{

Hand result = (Hand)super.clone();

return result;

}

public void sort()

{

Collections.sort(theCards);

}

}

public class Deck implements Cloneable{

private List<Card> cards = new ArrayList<Card>();

public Deck (){

for (Suit suit : Suit.values()) {

for (Rank rank : Rank.values()) {

cards.add(new Card(rank, suit));

}

}

}

public Card getCard (int index){

Card cr = new Card(cards.get(index).getRank(),cards.get(index).getSuit());

cards.remove(index);

return cr;

}

public Object clone()

{

Deck deck2 = new Deck ();

try{

for (int i=0 ; i < cards.size(); i++)

{

deck2.cards.add((Card)cards.get(i).clone());

}

} catch (CloneNotSupportedException e) {System.out.println("Clone error");}

return deck2;

}

public void shuffle()

{

Collections.shuffle(cards);

}

public void sort()

{

Collections.sort(cards, new Compare());

}

public String toString()

{

String s = "";

for(Card c : cards){

s+= c.toString();

}

return s;

}

public int search(Card c){

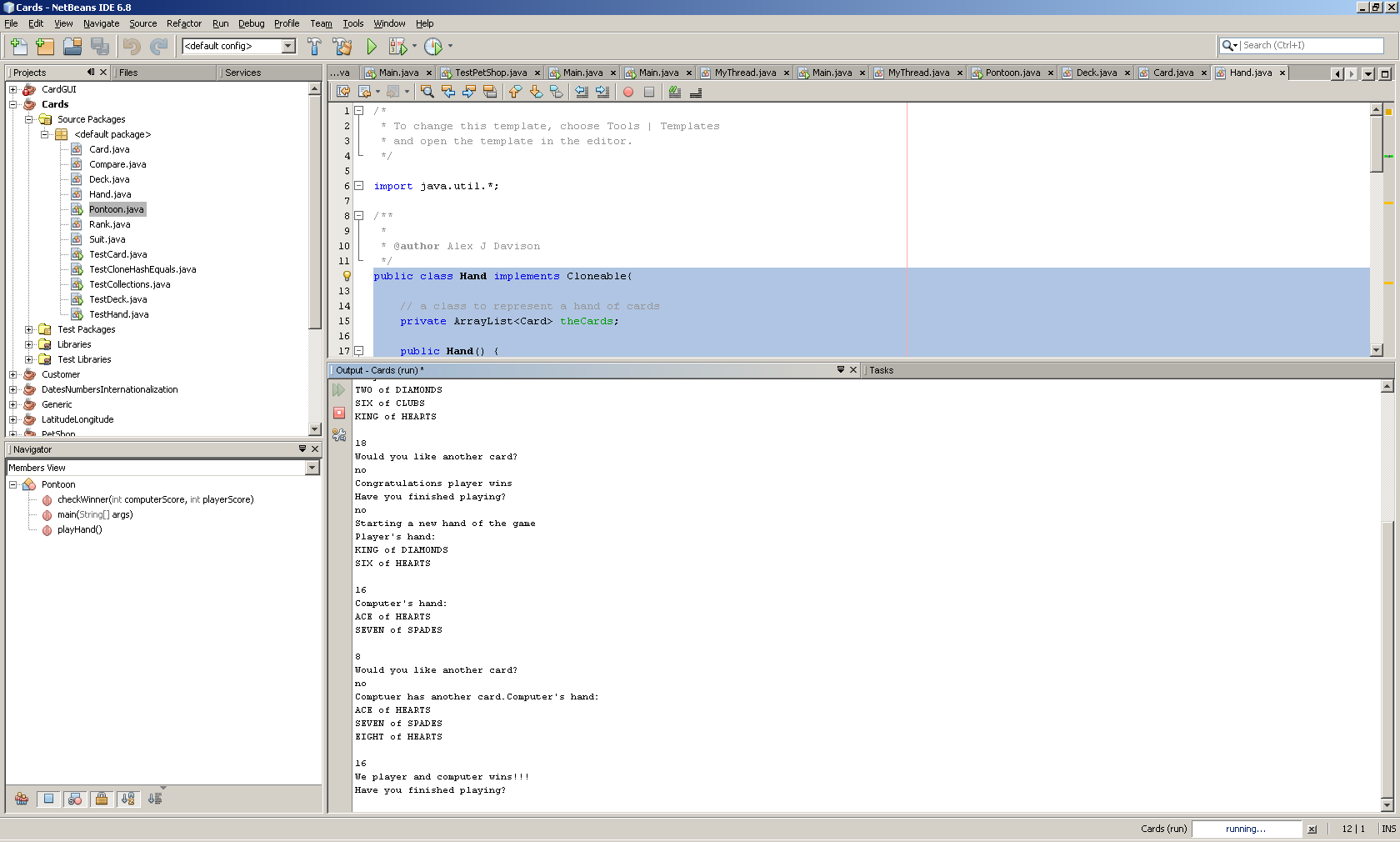
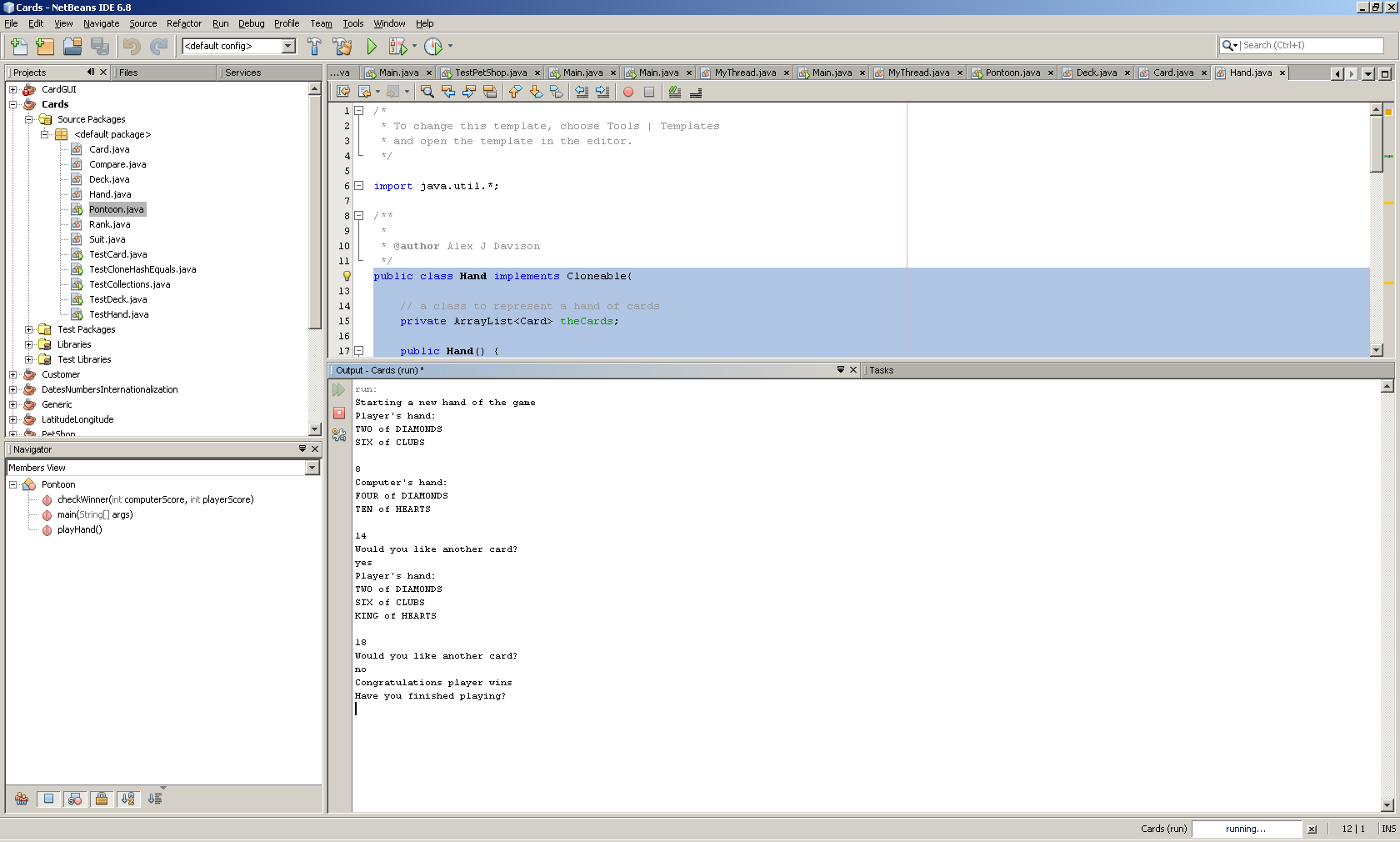
sort();

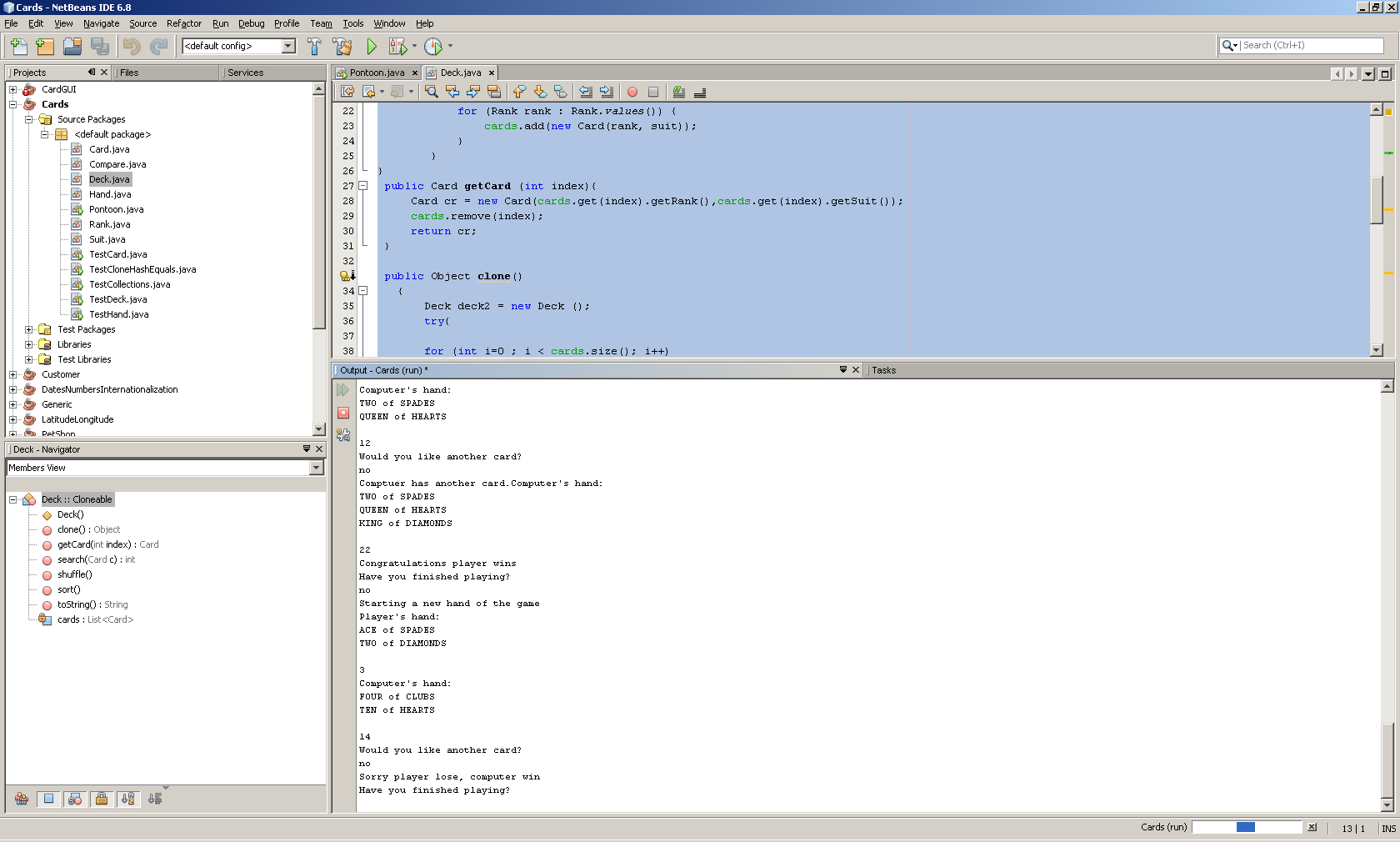
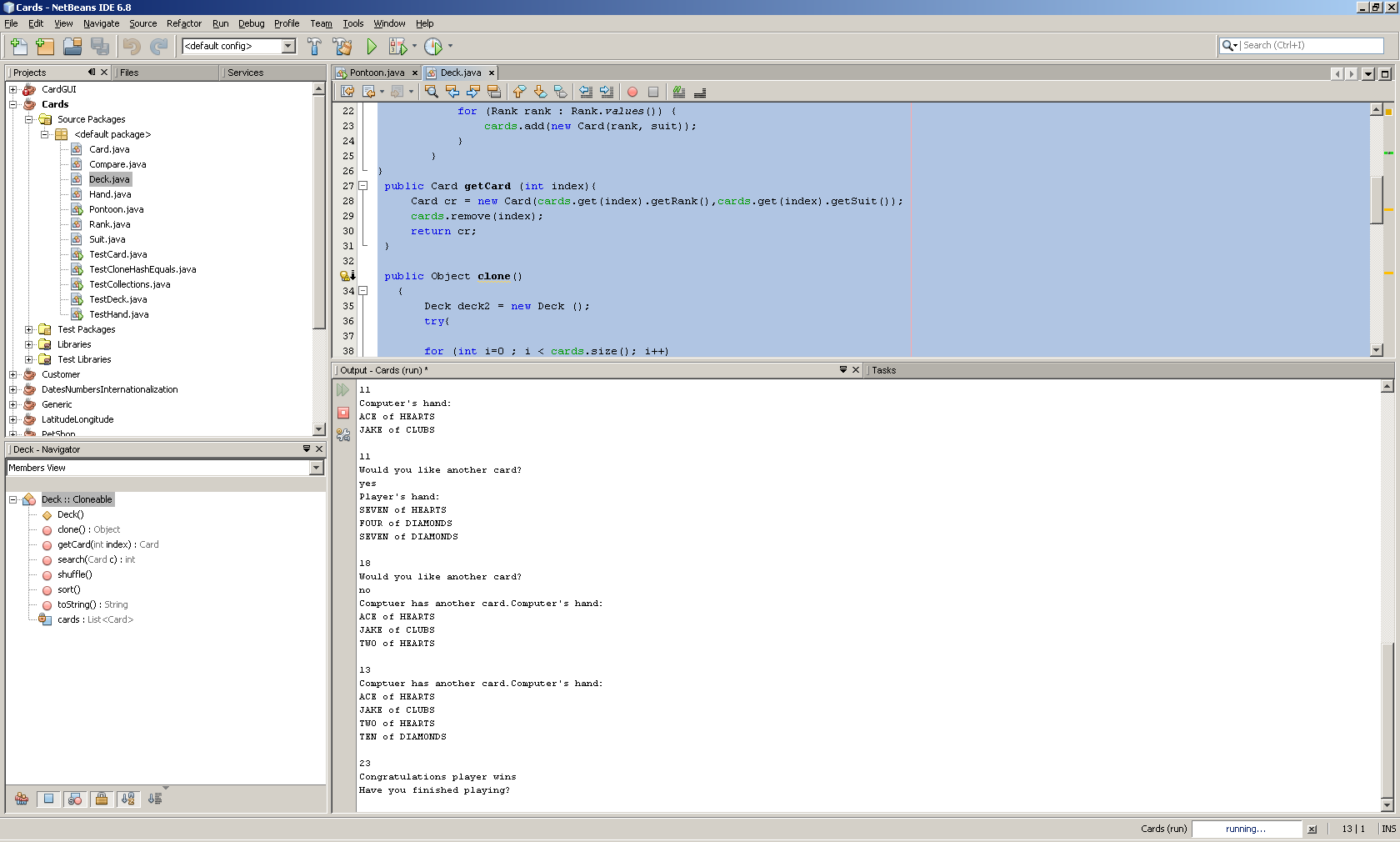
return Collections.binarySearch(cards, c);

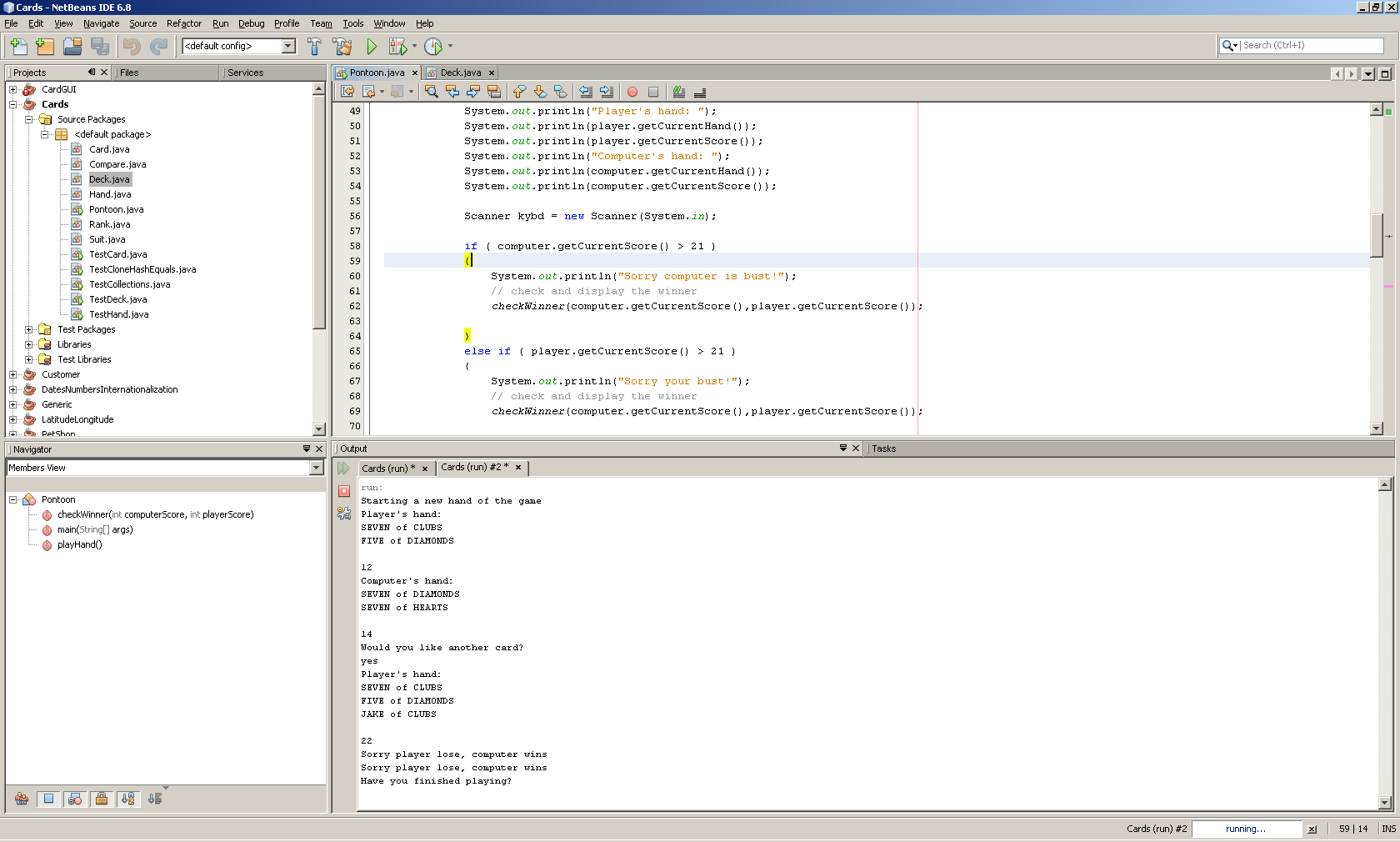
}

}

## Evidence (including screen shots) of testing your game







# Tutorial 11

## Code listing and evidence of testing for all exercises

### Question 1

#### Code

public class Buffer {

private int [] theData;

private final int MAX; // maximum size of the buffer

private int numItems;

private int head = 0;

private int tail = 0;

public Buffer (int m)

{

MAX = m;

theData = new int [MAX];

numItems = 0;

}

public synchronized int getNext() throws InterruptedException

{

while (numItems == 0) {

System.out.println("Buffer empty, need to wait");

wait();

}

int data = theData[head];

System.out.println(data + "removed from buffer: buffer now empty");

head = (++head) % MAX;

--numItems;

notifyAll();

return data;

}

public synchronized void add(int data) throws InterruptedException

{

while (numItems == MAX)

{

System.out.println("Buffer full please wait need to wait.");

wait();

}

System.out.println(data + "added to buffer: buffer contains" + numItems + "element");

theData[tail] = data;

tail = (++tail) % MAX;

++numItems;

notifyAll();

}

}

public class Consumer extends Thread implements Runnable {

private int number = 100;

private int num;

private Buffer buff;

public Consumer(Buffer b, int n)

{

buff = b;

num = n;

}

public void run(){

// the work of the thread takes place within the run method

// the thread terminates when the method ends

System.out.println("Consumer" + num + "attempting to get data to the buffer.");

try{

if (number ==buff.getNext())

{

System.out.println("Consumer" + num + "has retrevied" + number + "from buffer.");

}

}catch(InterruptedException e) {}

try{

Thread.sleep(100);

}catch (InterruptedException e) {}

}

}

public class Producer extends Thread implements Runnable {

private int number = 100;

private int num;

private Buffer buff;

public Producer(Buffer b, int n)

{

buff = b;

num = n;

}

public void run(){

// the work of the thread takes place within the run method

// the thread terminates when the method ends

System.out.println("Producer" + num + "attempting to add data to the buffer.");

try{

Thread.sleep(100);

}catch(InterruptedException e) {}

try{

buff.add(number);

}catch (InterruptedException e) {}

}

}

#### Test

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

Buffer b1 = new Buffer(10);

Consumer c1 = new Consumer(b1,1);

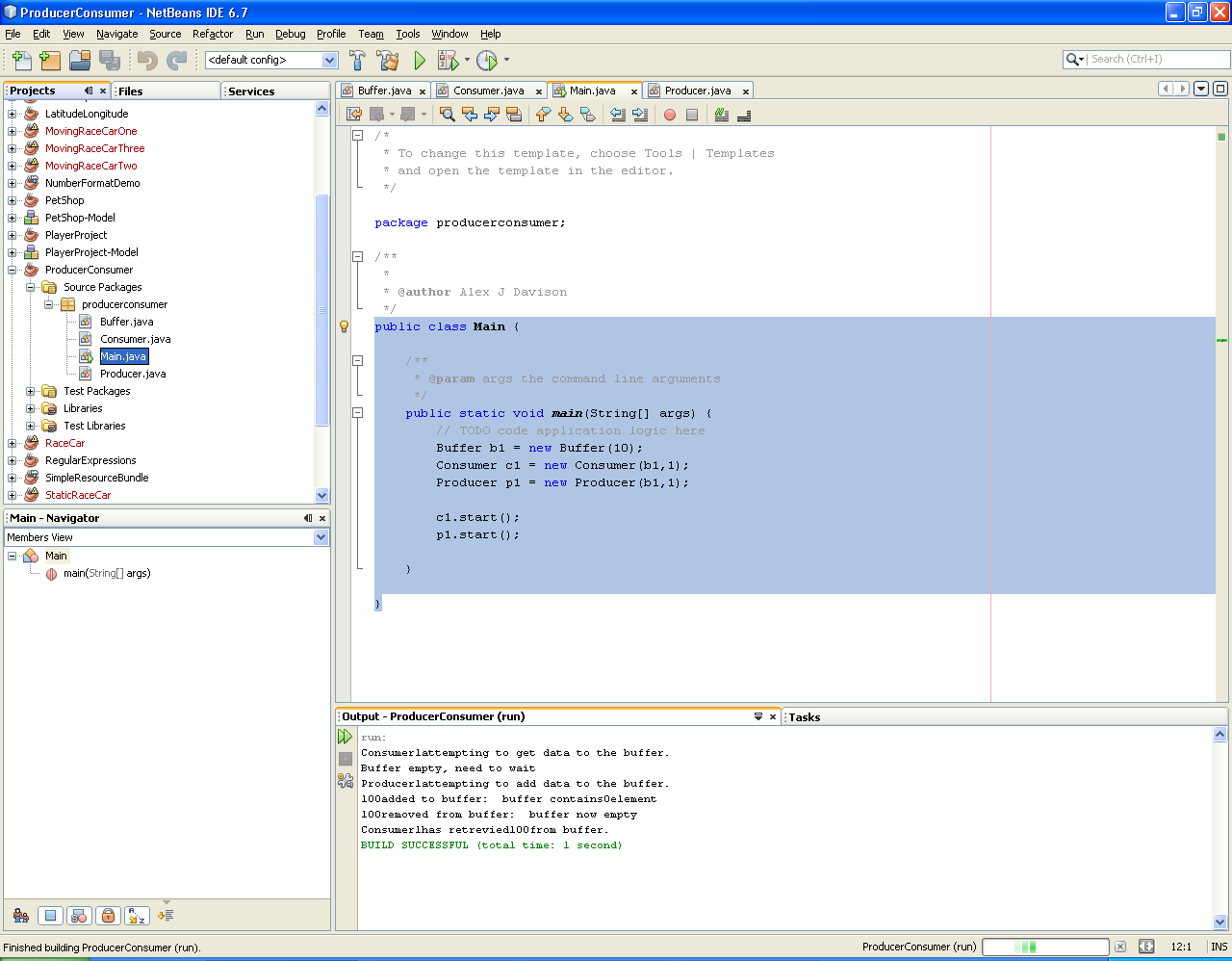
Producer p1 = new Producer(b1,1);

c1.start();

p1.start();

}

}



### Question 2/3/4

#### Code

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

int a;

a = inputInt("Enter Number of int type:");

System.out.println("Int entered:" + a);

double b;

b = inputDouble("Enter Number of double type:");

System.out.println("Double entered:" + b);

try{

b = divide(1,0);

System.out.println("Double entered:" + b);

}

catch (NumberZeroException e)

{

System.out.println("Can't divide by zero");

}

try{

b = divide(2,2);

System.out.println("Double entered:" + b);

}

catch (NumberZeroException e)

{

System.out.println("Can't divide by zero");

}

System.out.println(divide2(2,2));

System.out.println(divide2(1,0));

}

// TODO code application logic here

public static int inputInt(String prompt)

{

int num = 0;

boolean bol = true;

Scanner kybd = new Scanner(System.in);

while(bol == true)

{

try {

System.out.println(prompt);

num = kybd.nextInt();

bol = false;

}catch(InputMismatchException e){

kybd.next();

bol = true;

}

}

return num;

}

public static double inputDouble(String prompt)

{

double num = 0;

boolean bol = true;

Scanner kybd = new Scanner(System.in);

while(bol == true)

{

try {

System.out.println(prompt);

num = kybd.nextDouble();

bol = false;

}catch(InputMismatchException e){

kybd.next();

bol = true;

}

}

return num;

}

public static double divide (double num, double denum) throws NumberZeroException

{

if (denum == 0)

{

throw new NumberZeroException ();

}

return num/denum;

}

public static double divide2 (double num, double denum){

assert(!(denum == 0)): "error can't divide by zero";

return num/denum;

}

}

#### Test

