تمرین شماره 4

مکتب 25

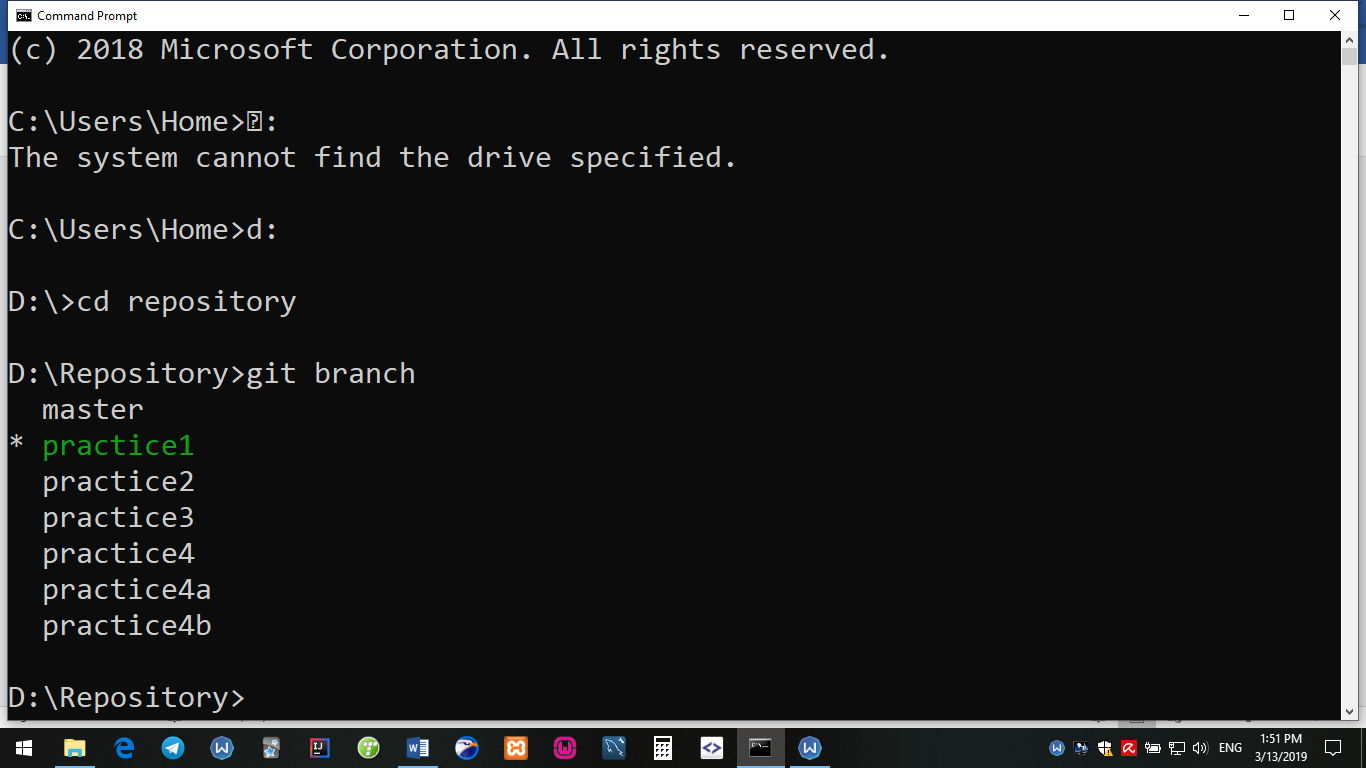
حمیدرضا ترابی

گیت:

ابتدا یک local repository ساختم: get init

سپس آدرس Remote repository ساخته شده در Github را clone کردم

برای هر سوال یک شاخه در نظر گرفتم:

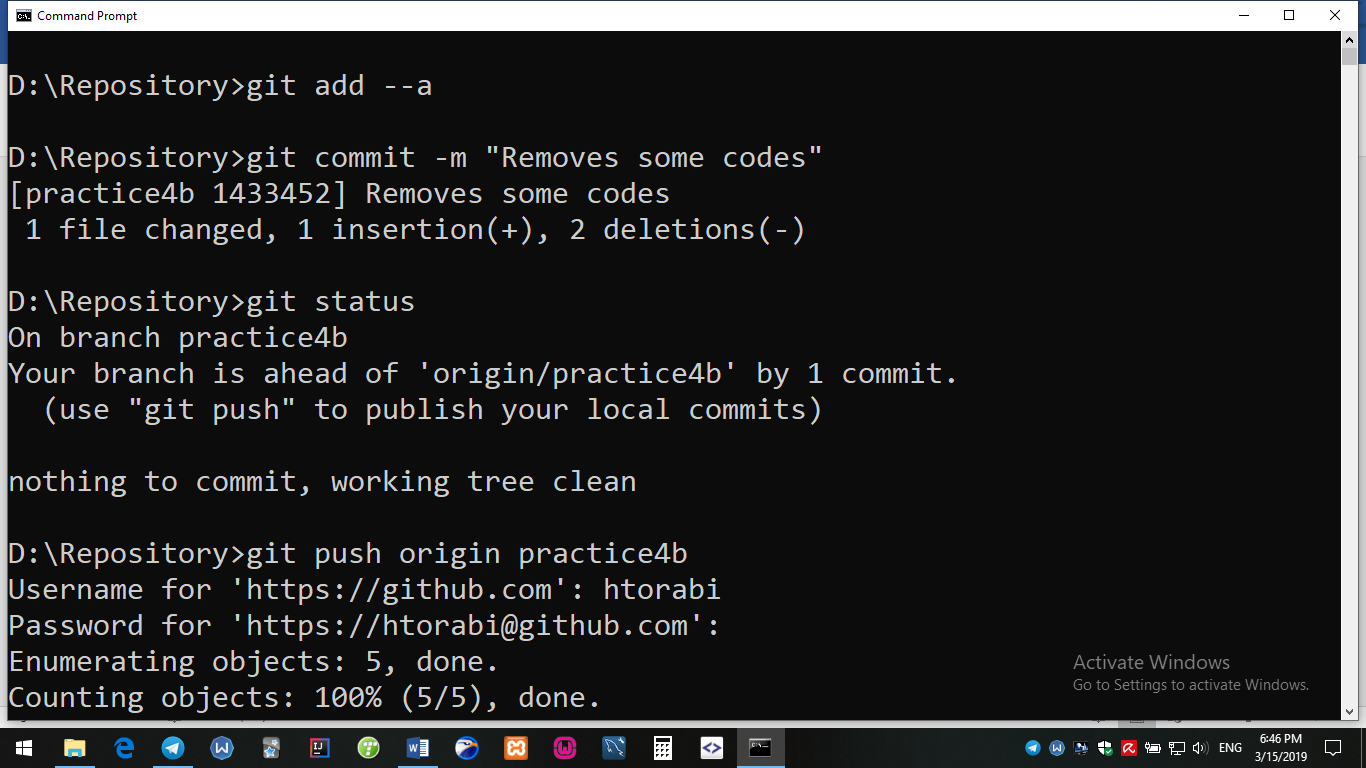


در هر یک از شاخه ها سوال موردنظر را حل کرده و کامیت کرده و به Repository پوش کردم

Git push origin master

برای مثال:

پس از تغییر در قسمت b سوال 4 که در شاخه practice4b موجود است ابتدا با دستور git add --a آن را به حالت stage بردیم و سپس با git commit -m “Removes some codes” آن را commit کردم و سپس آن را به origin پوش کردم:



دستور git status وضعیت گیت را نشان میدهد مثلا در شکل بالا تذکر میدهد کد شما از کد origin جلوتر است و باید پوش شود.

برای ساخت شاخه جدید دستور git branch <branch\_name>

حذف شاخه: git branch -d <branch\_name>

تغییر از یک شاخه به شاخه دیگر: git checkout <branch\_name>

آدرس Remote Repository :

<https://github.com/HTorabi/maktab25_hw4_torabi>

جاوا:

1. MessageCoder

کلاس MessageCoder را تشکیل دادیم که حروف پیام را به صورت کد در آورده و با یک عدد جمع میکند.

package com.maktab25.hw4.torabi.practice1;  
  
public class MessageCoder {  
 private String message;  
 private int code;  
  
 MessageCoder(String s, int code) {  
 message = s;  
 this.code = code;  
  
 }  
  
 public void setCode(int code) {  
 this.code = code;  
 }  
  
 public void setMessage(String message) {  
 this.message = message;  
 }  
  
 private String getMessage() {  
 return message;  
 }  
  
 public int getCode() {  
 return code;  
 }

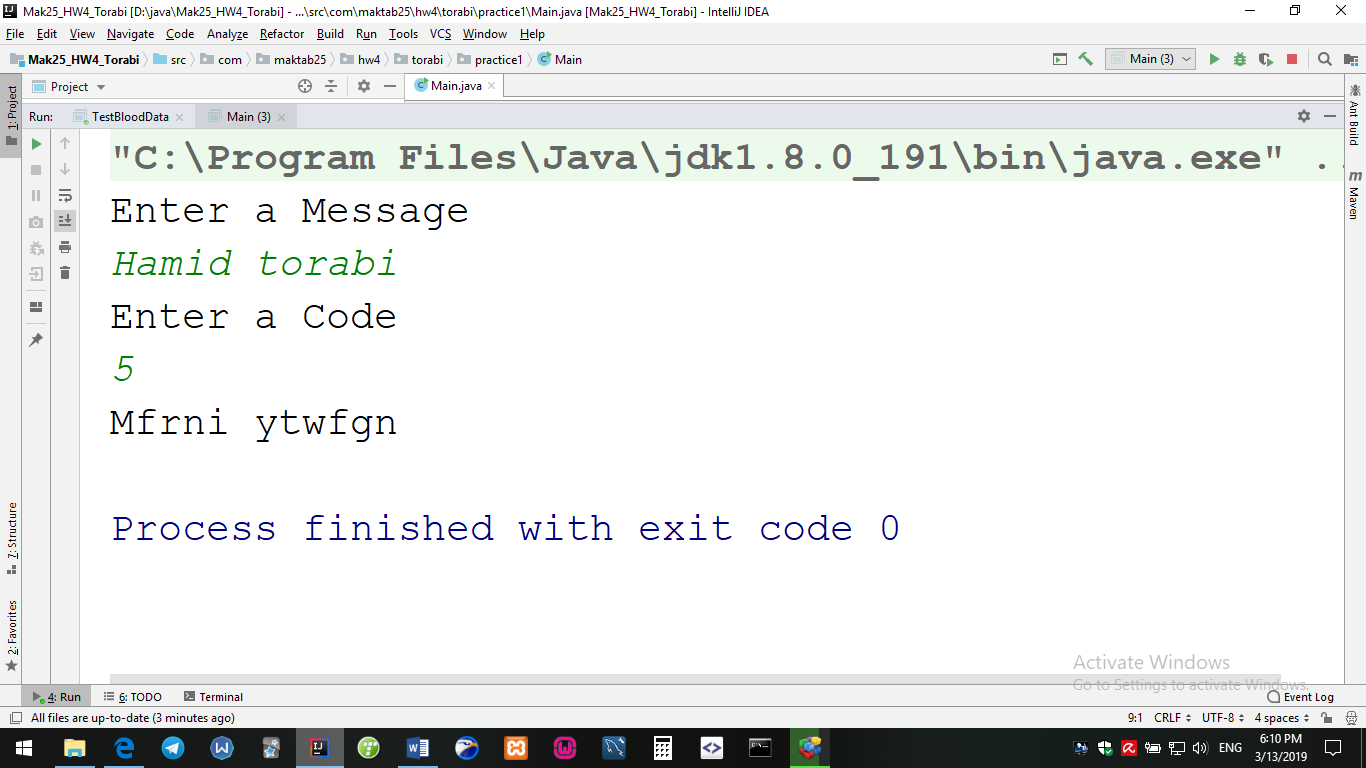
متد encrypt کاراکتر i ام message را به کد تبدیل می کند و متد encryptAll تمام پیام را.

private char encrypt(int i) {  
 char ch = ' ';  
 if (message.charAt(i) != ' ')  
 return (char) (((int) message.charAt(i)) + code);  
 else  
 return ch;  
  
 }  
  
 String encryptAll() {  
 char[] ch = new char[message.length()];  
 for (int i = 0; i < message.length(); i++)  
 ch[i] = encrypt(i);  
 return new String(ch);  
 }  
}

کلاس Main شامل متد main که پیام و کد را از ورودی میگیرد و با استفاده از کلاس MessageCoder رمزنگاری می کند.

package com.maktab25.hw4.torabi.practice1;  
import java.util.Scanner;  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner=new Scanner(System.in);  
 System.out.println("Enter a Message");  
 String message=scanner.nextLine();  
 System.out.println("Enter a Code");  
 int code=scanner.nextInt();  
 MessageCoder messageCoder = new MessageCoder(message,code);  
 System.out.println(messageCoder.encryptAll());  
 }  
}

خروجی با ورودی Hamid torabi و کد 5



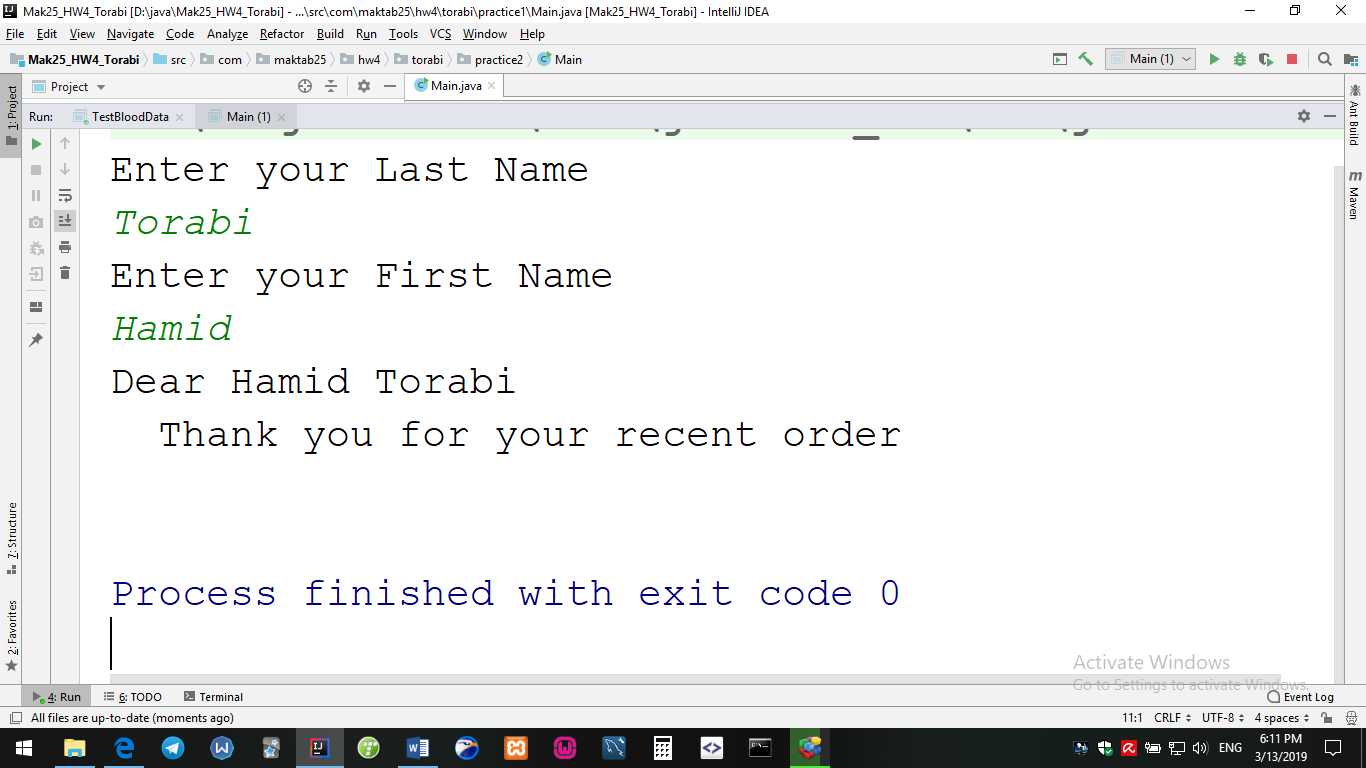
2. Form Letter Writer

کلاس FormLetterWriter را تعریف می کنیم که شامل دو متد است که ورودی آن فقط lastName یا firstName و lastName است و پیام هایی را متناسب با ورودی چاپ میکند.

package com.maktab25.hw4.torabi.practice2;  
  
public class FormLetterWriter {  
  
 static void displaySalutation(String lastName) {  
 System.out.println("Dear Mr. or Ms. " + lastName);  
 System.out.println(" Thank you for your recent order"+"\n");  
  
 }  
  
 static void displaySalutation(String firstName, String lastName) {  
 System.out.println("Dear " + firstName + " " + lastName);  
 System.out.println(" Thank you for your recent order"+"\n");  
 }  
}

کلاس Main شامل متد main که اسامی را گرفته و با استفاده از متد استاتیک displaySalutation در کلاس بالا پیام چاپ میکند.

package com.maktab25.hw4.torabi.practice2;  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println("Enter your Last Name");  
 String lastName = scanner.nextLine();  
 System.out.println("Enter your First Name");  
 String firstName = scanner.nextLine();  
 if (firstName.isEmpty())  
 FormLetterWriter.displaySalutation(lastName);  
 else  
 FormLetterWriter.displaySalutation(firstName, lastName);  
 }  
}



3. Billing

کلاس Billing شامل 3 متد متناسب با پارامترهای ورودی قیمت photo book و تعداد و هزینه کل را نمایش میدهد.

Package com.maktab25.hw4.torabi.practice3;  
  
public class Billing {  
 private static int photoBookPrice;  
  
 public static void setPhotoBookPrice(int photoBookPrice) {  
 com.maktab25.hw4.torabi.practice3.Billing.photoBookPrice = photoBookPrice;  
 }  
  
 public int getPhotoBookPrice() {  
 return photoBookPrice;  
 }  
  
 static void computeBill(String hotobook) {  
 System.out.println(“price of “+hotobook+” is: “ + photoBookPrice);  
 System.out.println(“total price:”+photoBookPrice \* (1 + 0.08));  
  
  
 }  
  
 static void computeBill(String hotobook, int quantity) {  
 System.out.println(“price of “+hotobook+” is:” + photoBookPrice + “\n” + “quantity:” + quantity);  
 System.out.println(“total price:”+photoBookPrice \* quantity \* (1 + 0.08));  
 }  
  
 static void computeBill(String hotobook, int quantity, int couponValue) {  
 System.out.println(“price of “+hotobook+” is:” + photoBookPrice + “\n” + “quantity:” + quantity + “\n” + “coupon value:” + couponValue);  
 System.out.println(“total price:”+ (photoBookPrice \* quantity-couponValue)\*1.08);  
 }  
}

کلاس Main که قیمت یک کتاب را به صورت استاتیک ثبت میکند. سپس ورودی های دیگر(نام کتاب، تعداد، و ارزش کوپن) را میگیرد و هزینه را چاپ میکند.

Package com.maktab25.hw4.torabi.practice3;  
  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println(“Enter Price of a Photo Book”);  
 int price = Integer.parseInt(scanner.nextLine());  
 Billing.setPhotoBookPrice(price);  
 System.out.println(“Enter name of Photo Book”);  
 String hotobook = scanner.nextLine();  
 System.out.println(“Enter number of the Photo Books”);  
 String quantityChecker = scanner.nextLine();  
 int quantity=0;  
 if (!quantityChecker.isEmpty())  
 quantity = Integer.parseInt(quantityChecker);  
 System.out.println(“Enter Coupon Value:if you don’t have any,press Enter”);  
 String couponChecker = scanner.nextLine();  
 int couponValue=0;  
 if (!couponChecker.isEmpty())  
 couponValue = Integer.parseInt(couponChecker);  
 if (quantityChecker.isEmpty())  
 Billing.computeBill(hotobook);  
 else if (couponChecker.isEmpty())  
 Billing.computeBill(hotobook, quantity);  
  
 else  
 Billing.computeBill(hotobook, quantity, couponValue);  
 }  
}

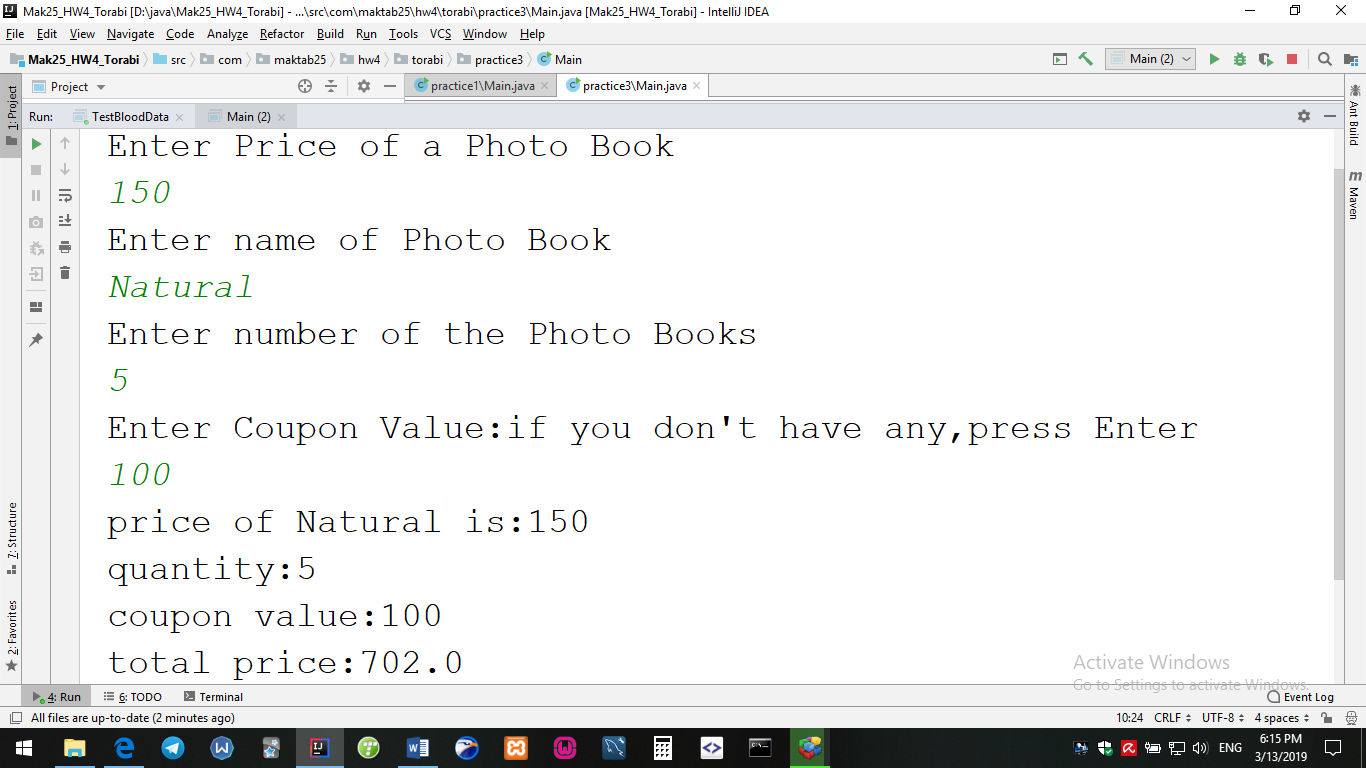
خروجی:

قیمت کتاب:150

اسم کتاب:Natural

تعداد:5

کوپن تخفیف:100



4. Blood Data & Patient

Blood Data .a

کلاس BloodData را تشکیل دادیم که شامل دو constructor (یکی default و دیگری با دو پارامتر) است

package com.maktab25.hw4.torabi.practice4;  
  
public class BloodData {  
 private String bloodType;  
 private char factor;  
  
 BloodData() {  
 bloodType = "O";  
 factor = '+';  
 }  
  
 BloodData(String bloodType, char factor) {  
 this.bloodType = bloodType.toUpperCase();  
  
 this.factor = factor;  
  
  
 }  
  
 public void setBloodType(String bloodType) {  
 this.bloodType = bloodType;  
 }  
  
 public void setFactor(char factor) {  
 this.factor = factor;  
 }  
  
 public char getFactor() {  
 return factor;  
 }  
  
 public String getBloodType() {  
 return bloodType;  
 }  
}

کلاس TestBloodData شامل main است که از ورودی گروه هونی و فاکتور را میگیرد وبررسی میکند گروه خونی و فاکتور valid است یا خیر.

Package com.maktab25.hw4.torabi.practice4;  
  
import java.util.Scanner;  
  
public class TestBloodData {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 String bloodType;  
 char factor;  
 System.out.println(“Enter Blood Type”);  
 while (true) {  
 bloodType = scanner.nextLine();  
 if ((bloodType.equalsIgnoreCase(“AB”) || bloodType.equalsIgnoreCase(“A”) || bloodType.equalsIgnoreCase(“B”) || bloodType.equalsIgnoreCase(“O”)))  
 break;  
 else  
 System.out.println(“This blood Type is not valid” + “\n” + “Enter another one”);  
 }  
 System.out.println(“Enter factor:+ or –“);  
 while (true) {  
 factor = scanner.nextLine().charAt(0);  
 if (factor == ‘+’ || factor == ‘-‘)  
 break;  
 else  
 System.out.println(“This blood factor is not valid” + “\n” + “Enter another one”);  
 }  
 BloodData bloodData = new BloodData(bloodType, factor);  
 System.out.println(bloodData.getBloodType() + bloodData.getFactor());  
  
 }  
}

4.b

کلاس Patient که یک Object از کلاس BloodData میسازد. و دو constructor در آن قرار دارد

package com.maktab25.hw4.torabi.practice4;  
  
public class Patient {  
 private int id;  
 private int age;  
 private BloodData bloodData = new BloodData();  
 static int MAX\_AGE;  
  
  
 Patient() {  
 id = 0;  
 age = 0;  
 bloodData.setBloodType("O");  
 bloodData.setFactor('+');  
 }  
  
 Patient(int id, int age, BloodData bloodData1) {  
 this.id = id;  
 this.age = age;  
 bloodData.setBloodType(bloodData1.getBloodType());  
 bloodData.setFactor(bloodData1.getFactor());  
 }  
  
 public BloodData getBloodData() {  
 return bloodData;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public int getId() {  
 return id;  
 }  
}

کلاس TestPatient مشخصات patient را میگیرد و پس از بررسی آن را نمایش میدهد.

package com.maktab25.hw4.torabi.practice4;  
  
import java.util.Scanner;  
  
public class TestPatient {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 Patient.MAX\_AGE = 120;  
 System.out.println("Enter ID");  
 int id = scanner.nextInt();  
 System.out.println("Enter age");  
 int age;  
 String bloodType;  
 char factor;  
 while (true) {  
 age = scanner.nextInt();  
 if (age > Patient.MAX\_AGE)  
 System.out.println("This is more than Max Age" + "\n" + "Enter another one");  
 else  
 break;  
 }  
 Scanner scanner1=new Scanner(System.in);  
 System.out.println("Enter Blood type");  
 while (true) {  
 bloodType = scanner1.nextLine();  
 if ((bloodType.equalsIgnoreCase("AB") || bloodType.equalsIgnoreCase("A") || bloodType.equalsIgnoreCase("B") || bloodType.equalsIgnoreCase("O")))  
 break;  
 System.out.println("This blood Type is not valid" + "\n" + "Enter another one");  
 }  
 System.out.println("Enter Blood factor");  
  
 while (true) {  
 factor = scanner1.nextLine().charAt(0);  
 if (factor == '+' || factor == '-')  
 break;  
 else  
 System.out.println("This blood factor is not valid" + "\n" + "Enter another one");  
 }  
 Patient patient = new Patient(id, age, new BloodData(bloodType, factor));  
 System.out.println("ID:" + patient.getId());  
 System.out.println("Age:" + patient.getAge());  
 System.out.println(patient.getBloodData().getBloodType() + patient.getBloodData().getFactor());  
  
  
 }  
}

خروجی برای ورودی های مختلف گروه خونی و سن بیشتر از حد مجاز:

