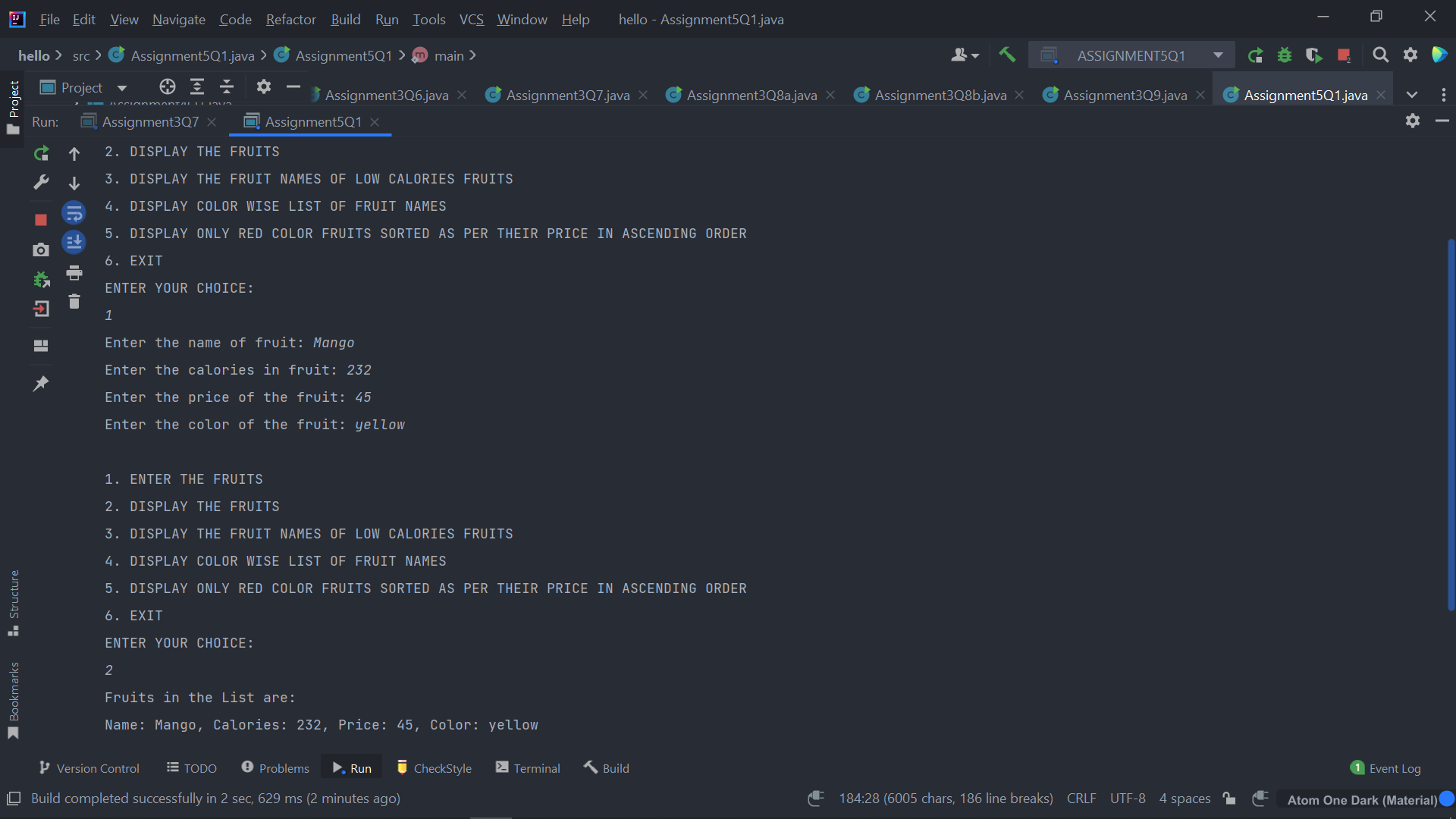
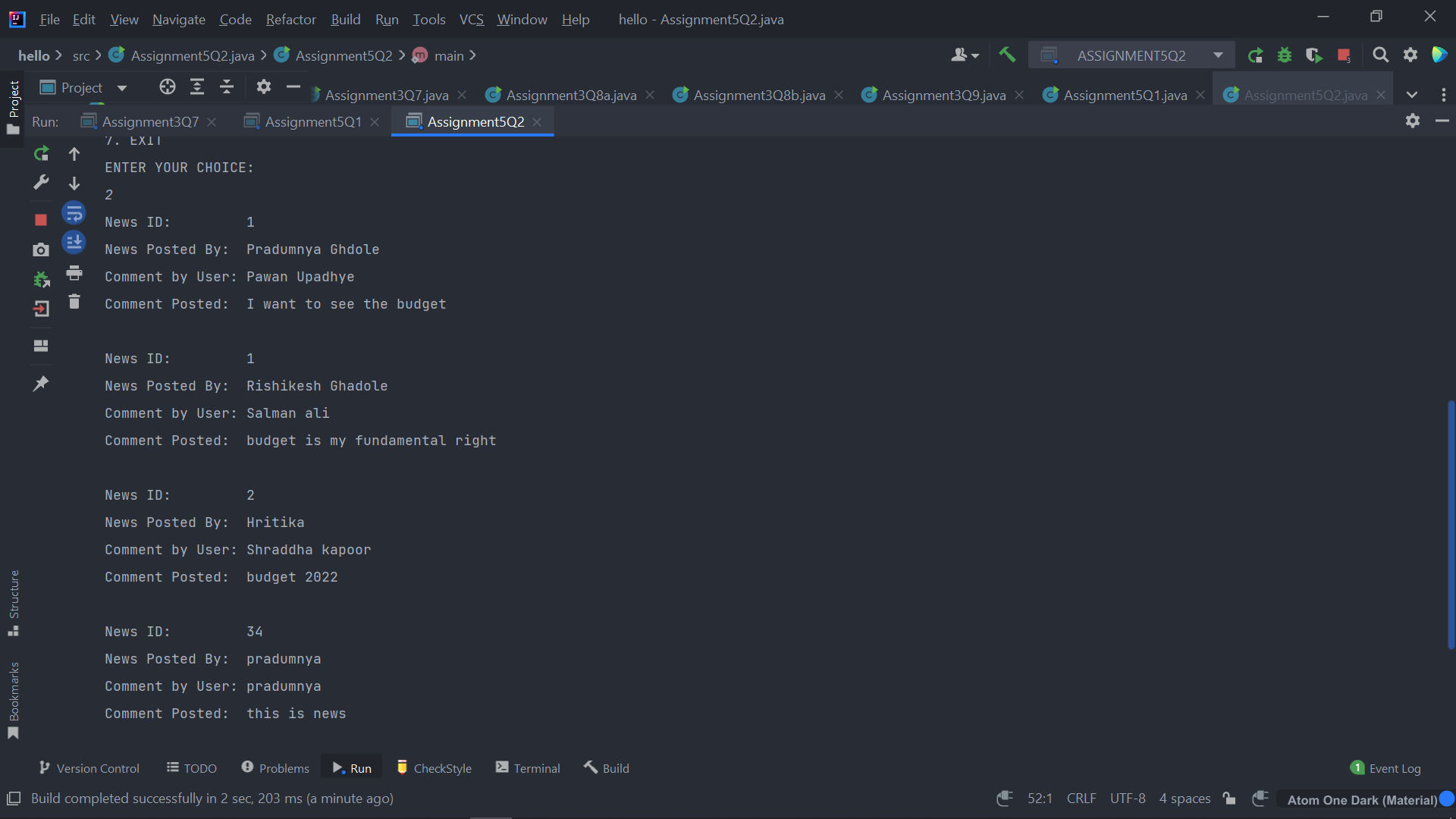
Q1.

*//1. Display the fruit names of low calories fruits i.e. calories < 100 sorted in descending order of calories.  
//2. Display color wise list of fruit names.  
// 3. Display only RED color fruits sorted as per their price in ascending order.  
  
  
import* java.util.\*;  
*import* java.util.stream.Collectors;  
*import* java.util.stream.*Stream*;  
  
*class* Fruits {  
 String name;  
 *int* calories;  
 *int* price;  
 String color;  
  
 *public* String getName() {  
 *return* name;  
 }  
  
 *public void* setName(String name) {  
 *this*.name = name;  
 }  
  
 *public int* getCalories() {  
 *return* calories;  
 }  
  
 *public void* setCalories(*int* calories) {  
 *this*.calories = calories;  
 }  
  
 *public int* getPrice() {  
 *return* price;  
 }  
  
 *public void* setPrice(*int* price) {  
 *this*.price = price;  
 }  
  
 *public* String getColor() {  
 *return* color;  
 }  
  
 *public void* setColor(String color) {  
 *this*.color = color;  
 }  
  
 @Override  
 *public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (!(o *instanceof* Fruits)) *return false*;  
 Fruits fruits = (Fruits) o;  
 *return* calories == fruits.calories && price == fruits.price && color.equals(fruits.color);  
 }  
  
 @Override  
 *public int* hashCode() {  
 *return* Objects.*hash*(calories, price, color);  
 }  
  
 *public static Comparator*<Fruits> *color1* = *new* Comparator<Fruits>() {  
 @Override  
 *public int* compare(Fruits o1, Fruits o2) {  
 *return* o1.getColor().compareTo(o2.getColor());  
 }  
 };  
  
 *public static Comparator*<Fruits> *calories1* = *new* Comparator<Fruits>() {  
 *public int* compare(Fruits f1,Fruits f2){  
 *return* f2.getCalories()-f1.getCalories();  
 }  
 };  
  
 *public static Comparator*<Fruits> *price1* = *new* Comparator<Fruits>() {  
 @Override  
 *public int* compare(Fruits o1, Fruits o2) {  
 *return* o1.getPrice()-o2.getPrice();  
 }  
 };  
}  
  
  
*public class* Assignment5Q1 {  
  
  
 *public static List*<String> reverseSort(ArrayList<Fruits> fruits) {  
 *List*<String> list = *new* ArrayList<>();  
 ArrayList<Fruits> fruits2 = *new* ArrayList<>();  
  
 fruits2 = (ArrayList<Fruits>) fruits.stream().filter((fruits1 -> {*return* fruits1.getCalories()<100;})).collect(Collectors.*toList*());  
  
 Collections.*sort*(fruits2,Fruits.*calories1*);  
  
 *for*(Fruits f: fruits2){  
 String string = f.getName();  
 list.add(string);  
 }  
  
 *return* list;  
 }  
 *public static* ArrayList<Fruits> sort(ArrayList<Fruits> fruits) {  
 Collections.*sort*(fruits,Fruits.*color1*);  
 *return* fruits;  
 }  
  
 *public static* ArrayList<Fruits> filterRedSortPrice(ArrayList<Fruits> fruits){  
 fruits = (ArrayList<Fruits>) fruits.stream().filter((fruits1 -> {*return* fruits1.getColor().toUpperCase().equals("RED");})).collect(Collectors.*toList*());  
 Collections.*sort*(fruits,Fruits.*price1*);  
 *return* fruits;  
 }  
  
 *public static void* main(String[] args) {  
  
 ArrayList<Fruits> arrayList = *new* ArrayList<>();  
 *int* choice;  
 Scanner sc = *new* Scanner(System.in);  
 *do* {  
 System.out.println("\n1. ENTER THE FRUITS");  
 System.out.println("2. DISPLAY THE FRUITS");  
 System.out.println("3. DISPLAY THE FRUIT NAMES OF LOW CALORIES FRUITS");  
 System.out.println("4. DISPLAY COLOR WISE LIST OF FRUIT NAMES");  
 System.out.println("5. DISPLAY ONLY RED COLOR FRUITS SORTED AS PER THEIR PRICE IN ASCENDING ORDER");  
 System.out.println("6. EXIT");  
 System.out.println("ENTER YOUR CHOICE: ");  
 choice = sc.nextInt();  
 *switch* (choice){  
  
 *case* 1:  
 Fruits fruits = *new* Fruits();  
 System.out.print("Enter the name of fruit: ");  
 String name = sc.next();  
 fruits.setName(name);  
 System.out.print("Enter the calories in fruit: ");  
 *int* calories = sc.nextInt();  
 fruits.setCalories(calories);  
 System.out.print("Enter the price of the fruit: ");  
 *int* price = sc.nextInt();  
 fruits.setPrice(price);  
 System.out.print("Enter the color of the fruit: ");  
 String color = sc.next();  
 fruits.setColor(color);  
  
 arrayList.add(fruits);  
 *break*;  
  
 *case* 2:  
 System.out.println("Fruits in the List are: ");  
 *for*(Fruits i: arrayList){  
 System.out.println("Name: "+i.getName()+", Calories: "+i.getCalories()+", Price: "+i.getPrice()+", Color: "+i.getColor());  
 }  
 *break*;  
  
 *case* 3: *List*<String> list = *new* ArrayList<>();  
  
 list = *reverseSort*(arrayList);  
  
 *for* (String str: list){  
 System.out.println("Fruit Name: "+str);  
 }  
 *break*;  
  
 *case* 4: ArrayList<Fruits> arrayList1 = *new* ArrayList<>();  
  
 arrayList1 = *sort*(arrayList);  
  
 *for*(Fruits fruits1: arrayList1){  
 System.out.println("Fruit Name: "+fruits1.getName()+", Fruit Color: "+fruits1.getColor());  
 }  
 *break*;  
  
 *case* 5: ArrayList<Fruits> arrayList2 = *new* ArrayList<>();  
 arrayList2 = *filterRedSortPrice*(arrayList);  
  
 *for* (Fruits fruits1: arrayList2){  
 System.out.println("Fruit Name: "+fruits1.getName()+", Fruit Price: "+fruits1.getPrice());  
 }  
 *break*;  
 *case* 6: System.*exit*(0);  
  
 *default*:  
 System.out.println("PLEASE ENTER THE RIGHT CHOICE!!");  
  
 }  
 }*while* (choice!=6);  
 }  
}



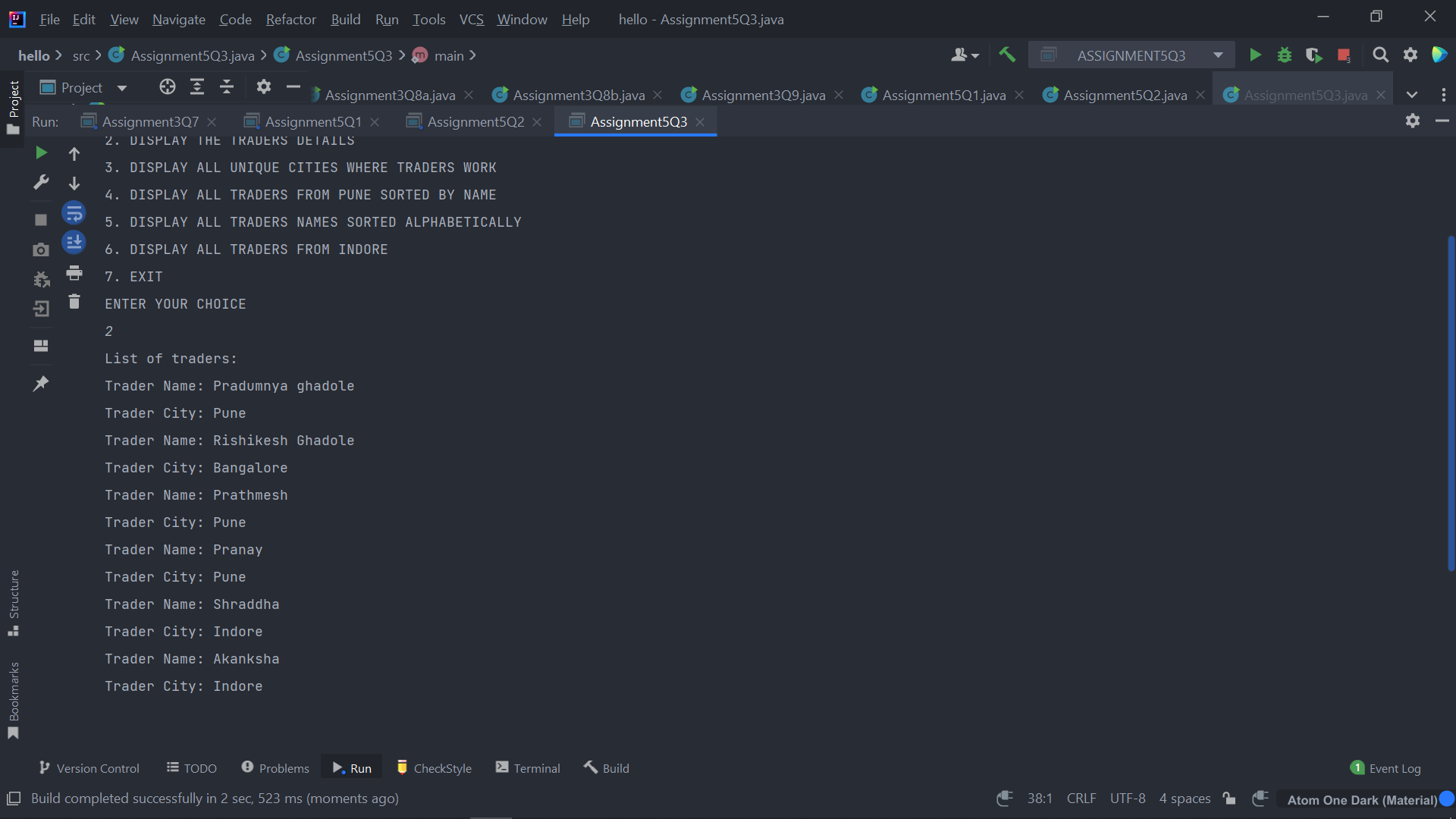
Q2.

*//1. Find out the newsId which has received maximum comments.  
//2. Find out how many times the word 'budget' arrived in user comments all news.  
//3. Find out which user has posted maximum comments.  
//4. Display commentByUser wise number of comments.  
  
  
import* java.util.\*;  
*import* java.util.stream.Collectors;  
*class* News {  
 *int* newsId;  
 String postedByUser;  
 String commentByUser;  
 String comment;  
  
 *public* News(*int* newsId, String postedByUser, String commentByUser, String comment) {  
 *this*.newsId = newsId;  
 *this*.postedByUser = postedByUser;  
 *this*.commentByUser = commentByUser;  
 *this*.comment = comment;  
 }  
  
 *public* News() {  
 }  
  
 *public int* getNewsId() {  
 *return* newsId;  
 }  
  
 *public void* setNewsId(*int* newsId) {  
 *this*.newsId = newsId;  
 }  
  
 *public* String getPostedByUser() {  
 *return* postedByUser;  
 }  
  
 *public void* setPostedByUser(String postedByUser) {  
 *this*.postedByUser = postedByUser;  
 }  
  
 *public* String getCommentByUser() {  
 *return* commentByUser;  
 }  
  
 *public void* setCommentByUser(String commentByUser) {  
 *this*.commentByUser = commentByUser;  
 }  
  
 *public* String getComment() {  
 *return* comment;  
 }  
  
 *public void* setComment(String comment) {  
 *this*.comment = comment;  
 }  
  
 @Override  
 *public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (!(o *instanceof* News)) *return false*;  
 News news = (News) o;  
 *return* newsId == news.newsId && commentByUser.equals(news.commentByUser) && comment.equals(news.comment);  
 }  
  
 @Override  
 *public int* hashCode() {  
 *return* Objects.*hash*(newsId, commentByUser, comment);  
 }  
  
}  
  
*public class* Assignment5Q2 {  
 *public static int* maxComments(*List*<News> news) {  
  
 *Map*<Integer,Integer> map = *new* HashMap<>();  
  
 *for* (News news1: news){  
 *int* id = news1.getNewsId();  
 *if*(map.containsKey(id)){  
 map.put(id,map.get(id)+1);  
 } *else* {  
 map.put(id,1);  
 }  
 }  
 *List*<*Map*.*Entry*<Integer, Integer>> list = *new* ArrayList<>(map.entrySet());  
 list.sort(*Map*.*Entry*.*comparingByValue*());  
 *return* list.get(list.size()-1).getKey();  
 }  
  
 *public static int* budgetCount (*List* < News > news) {  
 *int* count = 0;  
 *List*<String> list = *new* ArrayList<>();  
 *for*(News i: news){  
 String comment = i.getComment();  
 list.add(Arrays.*toString*(comment.split("budget")));  
 }  
  
 *return* list.size();  
 }  
  
 *public static* String maxCommentsByUser (*List* < News > news) {  
 *Map*<String,Integer> map = *new* HashMap<>();  
  
 *for*(News news1: news){  
 String username = news1.getCommentByUser();  
 *if*(map.containsKey(username)){  
 map.put(username,map.get(username)+1);  
 } *else* {  
 map.put(username,1);  
 }  
 }  
 *List*<*Map*.*Entry*<String, Integer>> list = *new* ArrayList<>(map.entrySet());  
 list.sort(*Map*.*Entry*.*comparingByValue*());  
 *return* list.get(list.size()-1).getKey();  
 }  
 *public static Map*<String, Integer> sortMaxCommentsByUser (*List* < News > news) {  
 *Map*<String,Integer> map = *new* HashMap<>();  
 *for*(News news1: news){  
 String username = news1.getCommentByUser();  
 *if*(map.containsKey(username)){  
 map.put(username,map.get(username)+1);  
 } *else* {  
 map.put(username,1);  
 }  
 }  
 HashMap<String, Integer> temp  
 = map.entrySet()  
 .stream()  
 .sorted((i1, i2)  
 -> i2.getValue().compareTo(  
 i1.getValue()))  
 .collect(Collectors.*toMap*(  
 *Map*.*Entry*::getKey,  
 *Map*.*Entry*::getValue,  
 (e1, e2) -> e1, LinkedHashMap::*new*));  
 *return* temp;  
 }  
  
 *public static void* main(String[] args) {  
  
 *List*<News> list = *new* ArrayList<>();  
 *int* choice;  
 Scanner sc = *new* Scanner(System.in);  
  
 News news1 = *new* News(1,  
 "Pradumnya Ghdole",  
 "Pawan Upadhye",  
 "I want to see the budget");  
  
 News news2 = *new* News(1,  
 "Rishikesh Ghadole",  
 "Salman ali",  
 "budget is my fundamental right");  
  
 News news3 = *new* News(2,  
 "Hritika",  
 "Shraddha kapoor",  
 "budget 2022");  
  
  
 list.add(news1);  
 list.add(news2);  
 list.add(news3);  
  
  
  
 *do*{  
 System.out.println("1. ENTER THE NEWS DETAILS");  
 System.out.println("2. DISPLAY THE NEWS DETAILS");  
 System.out.println("3. FIND OUT THE newsId WHICH HAS RECEIVED MAXIMUM COMMENTS.");  
 System.out.println("4. FIND OUT HOW MANY TIMES THE WORD 'BUDGET' ARRIVED IN USER COMMENTS ALL NEWS.");  
 System.out.println("5. FIND OUT WHICH USER HAS POSTED MAXIMUM COMMENTS.");  
 System.out.println("6. DISPLAY COMMENT BY USER WISE NUMBER OF COMMENTS.");  
 System.out.println("7. EXIT");  
 System.out.println("ENTER YOUR CHOICE:");  
 choice = sc.nextInt();  
  
 *switch* (choice){  
 *case* 1:  
 News news = *new* News();  
  
 *try* {  
 System.out.print("Enter the newsID: ");  
 *int* newId = sc.nextInt();  
 news.setNewsId(newId);  
  
 sc.nextLine();  
 System.out.print("Enter the name who posted the news: ");  
 String name = sc.nextLine();  
 news.setPostedByUser(name);  
  
  
 System.out.print("Enter the comment: ");  
 String comment = sc.nextLine();  
 news.setComment(comment);  
  
 System.out.print("Enter the name who posted the comment: ");  
 String username = sc.nextLine();  
 news.setCommentByUser(username);  
  
 list.add(news);  
  
 }*catch* (Exception e){  
 System.out.println(e);  
 System.out.println("Enter the correct input please!!");  
 }  
 *break*;  
  
 *case* 2: *for*(News new1: list){  
 System.out.println("News ID: "+new1.getNewsId());  
 System.out.println("News Posted By: "+new1.getPostedByUser());  
 System.out.println("Comment by User: "+new1.getCommentByUser());  
 System.out.println("Comment Posted: "+new1.getComment());  
 System.out.println();  
 }  
 *break*;  
  
 *case* 3: *int* id = *maxComments*(list);  
 System.out.println("NEWS ID WHICH HAS RECEIVED MAXIMUM COMMENTS: "+id);  
 System.out.println();  
 *break*;  
  
 *case* 4: *int* countBudget = *budgetCount*(list);  
 System.out.println("HOW MANY TIMES THE WORD 'BUDGET' ARRIVED IN USER COMMENTS ALL NEWS: "+countBudget);  
 System.out.println();  
 *break*;  
  
 *case* 5: String name = *maxCommentsByUser*(list);  
 System.out.println("USER HAS POSTED MAXIMUM COMMENTS: "+name);  
 System.out.println();  
 *break*;  
  
 *case* 6: *Map*<String,Integer> maxCommentByUser = *sortMaxCommentsByUser*(list);  
  
 *for* (*Map*.*Entry*<String,Integer> mp: maxCommentByUser.entrySet()){  
 System.out.println("Name of the User: "+mp.getKey());  
 System.out.println("Numbers of Comments: "+mp.getValue());  
 System.out.println();  
 }  
 System.out.println();  
 *break*;  
 }  
 }*while* (choice!=7);  
 }  
}



Q3.

*import* java.util.\*;  
  
*class* Trader {  
 String name;  
 String city;  
  
 *public* Trader(String name, String city) {  
 *this*.name = name;  
 *this*.city = city;  
 }  
 *public* Trader(){  
  
 }  
  
 *public* String getName() {  
 *return* name;  
 }  
  
 *public void* setName(String name) {  
 *this*.name = name;  
 }  
  
 *public* String getCity() {  
 *return* city;  
 }  
  
 *public void* setCity(String city) {  
 *this*.city = city;  
 }  
  
 @Override  
 *public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (!(o *instanceof* Trader)) *return false*;  
 Trader trader = (Trader) o;  
 *return* name.equals(trader.name) && city.equals(trader.city);  
 }  
  
 @Override  
 *public int* hashCode() {  
 *return* Objects.*hash*(name, city);  
 }  
  
 *public static Comparator*<Trader> *comparator* = *new* Comparator<Trader>() {  
 @Override  
 *public int* compare(Trader o1, Trader o2) {  
 *return* o1.getName().compareTo(o2.getName());  
 }  
 };  
}  
  
  
*public class* Assignment5Q3 {  
  
 *public static List*<String> printUniqueCities (*List* <Trader> traders) {  
  
 *Map*<String,String> map = *new* HashMap<>();  
 *List*<String> cities = *new* ArrayList<>();  
  
 *for* (Trader trader: traders){  
 map.putIfAbsent(trader.getCity(),trader.getName());  
 }  
  
 *for* (*Map*.*Entry*<String,String> mp: map.entrySet()){  
 cities.add(mp.getKey());  
 }  
 *return* cities;  
 }  
  
 *public static List*<String> tradersFromPuneSortByName(*List*<Trader> traders) {  
 *List*<String> tradersFromPune = *new* ArrayList<>();  
  
 *for* (Trader trader: traders){  
 *if*(trader.getCity().toUpperCase().equals("PUNE")){  
 tradersFromPune.add(trader.getName());  
 }  
 }  
  
 Collections.*sort*(tradersFromPune);  
 *return* tradersFromPune;  
 }  
 *public static List*<Trader> allTrader3Names(*List*<Trader> traders) {  
 traders.sort(Trader.*comparator*);  
 *return* traders;  
 }  
 *public static* ArrayList<Trader> areAnyTradersFromIndore(ArrayList<Trader> traders) {  
 ArrayList<Trader> traderList = *new* ArrayList<>();  
  
 *for*(Trader trader: traders){  
 *if*(trader.getCity().toUpperCase().equals("INDORE")){  
 traderList.add(trader);  
 }  
 }  
 *return* traderList;  
 }  
  
 *public static void* main(String[] args) {  
 Scanner sc = *new* Scanner(System.in);  
  
 ArrayList<Trader> arrayList = *new* ArrayList<>();  
  
 Trader trader1 = *new* Trader("Pradumnya ghadole","Pune");  
 Trader trader2 = *new* Trader("Rishikesh Ghadole","Bangalore");  
 Trader trader3 = *new* Trader("Prathmesh","Pune");  
 Trader trader4 = *new* Trader("Pranay","Pune");  
 Trader trader5 = *new* Trader("Shraddha ","Indore");  
 Trader trader6 = *new* Trader("Akanksha ","Indore");  
  
 arrayList.add(trader1);  
 arrayList.add(trader2);  
 arrayList.add(trader3);  
 arrayList.add(trader4);  
 arrayList.add(trader5);  
 arrayList.add(trader6);  
  
 *int* choice;  
 *do*{  
 System.out.println("\n1. ENTER THE TRADERS DETAILS");  
 System.out.println("2. DISPLAY THE TRADERS DETAILS");  
 System.out.println("3. DISPLAY ALL UNIQUE CITIES WHERE TRADERS WORK");  
 System.out.println("4. DISPLAY ALL TRADERS FROM PUNE SORTED BY NAME");  
 System.out.println("5. DISPLAY ALL TRADERS NAMES SORTED ALPHABETICALLY");  
 System.out.println("6. DISPLAY ALL TRADERS FROM INDORE");  
 System.out.println("7. EXIT");  
 System.out.println("ENTER YOUR CHOICE");  
 choice = sc.nextInt();  
  
 *switch* (choice){  
 *case* 1:  
 Trader trader = *new* Trader();  
 sc.nextLine();  
 System.out.println("Enter the Trader Name: ");  
 String name = sc.nextLine();  
 trader.setName(name);  
  
 System.out.println("Enter the City Name: ");  
 String city = sc.nextLine();  
 trader.setCity(city);  
  
 arrayList.add(trader);  
 *break*;  
  
 *case* 2:  
 System.out.println("List of traders: ");  
 *for* (Trader trade: arrayList){  
 System.out.println("Trader Name: "+trade.getName());  
 System.out.println("Trader City: "+trade.getCity());  
 }  
 System.out.println();  
 *break*;  
  
 *case* 3:  
 *List*<String> list = *new* ArrayList<>();  
 list = *printUniqueCities*(arrayList);  
  
 System.out.println("All uniques cities where traders work: ");  
 *for* (String str: list){  
 System.out.println(str);  
 }  
 System.out.println();  
 *break*;  
  
 *case* 4:  
 *List*<String> list1 = *new* ArrayList<>();  
 list1 = *tradersFromPuneSortByName*(arrayList);  
  
 System.out.println("All traders from Pune: ");  
 *for* (String str: list1){  
 System.out.println(str);  
 }  
 System.out.println();  
 *break*;  
  
 *case* 5: *List*<Trader> arraylist2 = *new* ArrayList<>();  
 arraylist2 = *allTrader3Names*(arrayList);  
  
 System.out.println("All Traders Name:");  
 *for*(Trader trader7: arraylist2){  
 System.out.println("Name: "+trader7.getName());  
 System.out.println("City: "+trader7.getCity());  
 System.out.println();  
 }  
 System.out.println();  
 *break*;  
  
 *case* 6: *List*<Trader> traderList = *new* ArrayList<>();  
 traderList = *areAnyTradersFromIndore*(arrayList);  
 System.out.println("All traders from Indore: ");  
 *for*(Trader trader7: traderList){  
 System.out.println("Name: "+trader7.getName());  
 }  
 System.out.println();  
 *break*;  
  
  
 *case* 7: System.*exit*(0);  
  
 *default*:  
 System.out.println("PLEASE ENTER THE RIGHT CHOICE!");  
 }  
 }*while* (choice!=7);  
 }  
}



Q4.

*import* java.util.\*;  
*import* java.util.stream.Collectors;  
  
*class* Trader1 {  
 String name;  
 String city;  
  
 *public* Trader1(String name, String city) {  
 *this*.name = name;  
 *this*.city = city;  
 }  
 *public* Trader1(){  
  
 }  
 *public* String getName() {  
 *return* name;  
 }  
  
 *public void* setName(String name) {  
 *this*.name = name;  
 }  
  
 *public* String getCity() {  
 *return* city;  
 }  
  
 *public void* setCity(String city) {  
 *this*.city = city;  
 }  
  
 @Override  
 *public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (!(o *instanceof* Trader1)) *return false*;  
 Trader1 trader1 = (Trader1) o;  
 *return* name.equals(trader1.name) && city.equals(trader1.city);  
 }  
  
 @Override  
 *public int* hashCode() {  
 *return* Objects.*hash*(name, city);  
 }  
}  
  
*class* Transaction {  
 Trader1 trader1;  
 *int* year;  
 *int* value;  
  
 *public* Transaction(){}  
 *public* Transaction(Trader1 trader1, *int* year, *int* value) {  
 *this*.trader1 = trader1;  
 *this*.year = year;  
 *this*.value = value;  
 }  
  
 *public* Trader1 getTrader1() {  
 *return* trader1;  
 }  
  
 *public void* setTrader1(Trader1 trader1) {  
 *this*.trader1 = trader1;  
 }  
  
 *public int* getYear() {  
 *return* year;  
 }  
  
 *public void* setYear(*int* year) {  
 *this*.year = year;  
 }  
  
 *public int* getValue() {  
 *return* value;  
 }  
  
 *public void* setValue(*int* value) {  
 *this*.value = value;  
 }  
  
 @Override  
 *public boolean* equals(Object o) {  
 *if* (*this* == o) *return true*;  
 *if* (!(o *instanceof* Transaction)) *return false*;  
 Transaction that = (Transaction) o;  
 *return* year == that.year && value == that.value && trader1.equals(that.trader1);  
 }  
  
 @Override  
 *public int* hashCode() {  
 *return* Objects.*hash*(trader1, year, value);  
 }  
  
 *public static Comparator*<Transaction> *MaximumValue* = *new* Comparator<Transaction>() {  
 @Override  
 *public int* compare(Transaction o1, Transaction o2) {  
 *return* o1.getValue()-o2.getValue();  
 }  
 };  
}  
*public class* Assignment5Q4 {  
 *public static List*<Transaction> sortTransactions(*List* <Transaction> transactions) {  
  
 transactions = transactions.stream().filter((t1)->t1.getYear()==2011).collect(Collectors.*toList*());  
  
 transactions.sort(Transaction.*MaximumValue*);  
 *return* transactions;  
  
 }  
  
 *public static List*<Integer> transactionsValuesDelhi(*List*<Transaction> transactions) {  
 *List*<Integer> transactionValue = *new* ArrayList<>();  
  
 transactions = transactions.stream().filter((t1)->t1.getTrader1().getCity().toUpperCase().equals("DELHI")).collect(Collectors.*toList*());  
  
 *for*(Transaction t1: transactions){  
 transactionValue.add(t1.getValue());  
 }  
 *return* transactionValue;  
 }  
 *public static int* highestTransaction(*List*<Transaction> transactions){  
 *int* MaxTransaction;  
 MaxTransaction = transactions.stream().mapToInt(Transaction::getValue).max().orElseThrow(NoSuchElementException::*new*);  
 *return* MaxTransaction;  
 }  
  
 *public static int* smallestTransaction(*List*<Transaction> transactions){  
 *return* transactions.stream().mapToInt(Transaction::getValue).min().orElseThrow(NoSuchElementException::*new*);  
 }  
 *public static void* main(String[] args) {  
 *List*<Transaction> arraylist = *new* ArrayList<>();  
  
 Scanner sc = *new* Scanner(System.in);  
 *int* choice;  
  
 *do*{  
 System.out.println("1. ENTER THE DETAILS OF TRADERS & THEIR TRANSACTIONS");  
 System.out.println("2. DISPLAY THE DETAILS OF TRADERS & THEIR TRANSACTION");  
 System.out.println("3. DISPLAY ALL TRANSACTIONS IN THE YEAR 2011 IN SORTED ORDER");  
 System.out.println("4. DISPLAY ALL TRANSACTIONS FROM TRADERS WHO LIVES IN DELHI");  
 System.out.println("5. DISPLAY THE HIGHEST VALUE OF ALL THE TRANSACTIONS");  
 System.out.println("6. DISPLAY THE SMALLEST VALUE OF ALL THE TRANSACTIONS");  
 System.out.println("7. EXIT");  
 System.out.println("ENTER YOUR CHOICE");  
 choice = sc.nextInt();  
 *switch* (choice){  
 *case* 1:  
 Transaction transaction = *new* Transaction();  
 Trader1 trader1 = *new* Trader1();  
 sc.nextLine();  
 System.out.print("ENTER THE NAME OF THE TRADER: ");  
 String name = sc.nextLine();  
 trader1.setName(name);  
  
 System.out.print("ENTER THE CITY NAME OF THE TRADER: ");  
 String city = sc.nextLine();  
 trader1.setCity(city);  
  
 transaction.setTrader1(trader1);  
  
 System.out.print("ENTER THE YEAR OF TRANSACTION: ");  
 *int* year = sc.nextInt();  
 transaction.setYear(year);  
  
 System.out.print("ENTER THE AMOUNT OF TRANSACTION: ");  
 *int* amount = sc.nextInt();  
 transaction.setValue(amount);  
  
 arraylist.add(transaction);  
 *break*;  
  
 *case* 2:  
 System.out.println("List of all Traders and their Transactions");  
  
 *for* (Transaction t: arraylist){  
 System.out.println("Trader Name: "+t.getTrader1().getName());  
 System.out.println("Trader City: "+t.getTrader1().getCity());  
 System.out.println("Year: "+t.getYear());  
 System.out.println("Value: "+t.getValue());  
 System.out.println();  
 }  
 *break*;  
  
 *case* 3:  
 System.out.println("All the transactions in the year 2011 in sorted order: ");  
 *List*<Transaction> transactionList = *new* ArrayList<>();  
 transactionList = *sortTransactions*(arraylist);  
  
 *for* (Transaction t1: transactionList){  
 System.out.println("Trader Name: "+t1.getTrader1().getName());  
 System.out.println("Trader City: "+t1.getTrader1().getCity());  
 System.out.println("Value: "+t1.getValue());  
 System.out.println("Year: "+t1.getYear());  
 System.out.println();  
 }  
 *break*;  
  
 *case* 4:  
 System.out.println("All transactions from the traders lives in Delhi: ");  
 *List*<Integer> list = *new* ArrayList<>();  
 list = *transactionsValuesDelhi*(arraylist);  
  
 list.forEach(System.out::println);  
 *break*;  
  
 *case* 5: *int* Maxvalue = *highestTransaction*(arraylist);  
 System.out.println("Highest Transaction value: "+Maxvalue);  
 *break*;  
  
 *case* 6: *int* Minvalue = *smallestTransaction*(arraylist);  
 System.out.println("Lowest Transaction value: "+Minvalue);  
 *break*;  
  
 *case* 7: System.*exit*(0);  
  
 *default*:  
 System.out.println("PLEASE ENTER THE CORRECT CHOICE: ");  
  
 }  
 }*while* (choice!=7);  
  
  
 }  
}

