1. *TPSP.java*

package TPSP16S2;  
  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.Map;  
import java.util.Map.Entry;  
  
import Entity.Card;  
import Entity.Constants;  
import Service.CardService;  
import Service.InstructionService;  
  
public class TPSP {  
   
 public static void main(String[] args){  
   
 Constants cons = new Constants(); // set up attractions  
// System.out.println("-----------------Theme Park Smart-Pass System-----------------");  
   
 if(args.length!=4){  
 System.*out*.println("ERROR!Please input right arguments ordered by card file, instruction file, result file and report file.");  
 }else{  
 String cardFile = args[0];  
 String instructionFile = args[1];  
 String resultFile = args[2];  
 String reportFile = args[3];  
   
 HashMap<String,Card> cardMap = new HashMap<String,Card>();  
   
 CardService cardService = new CardService();  
 cardMap = cardService.readCardFile(cardFile);  
  
   
 InstructionService instructionService = new InstructionService();  
 instructionService.readInstrcutionFile(instructionFile, cardMap, reportFile);  
  
   
 cardService.writeCardIntoResultFile(resultFile, cardMap);  
 }  
  
 }  
}

1. *Constants.java*

package Entity;  
  
import java.util.HashMap;  
  
*/\*\*  
 \* Initialize Attractions  
 \** ***@version*** *1.0  
 \*  
 \*/*public class Constants {  
  
 public static final HashMap<String,Attraction> *attracMap*= new HashMap<String,Attraction>();  
   
 public Constants(){  
 Attraction spidermanEscape = new Attraction();  
 spidermanEscape.setAttractType("Thrill Rides");  
 spidermanEscape.setAttractName("Spiderman Escape");  
 spidermanEscape.setAge(">=8");  
 spidermanEscape.setHeight(">=100");  
 *attracMap*.put("Spiderman Escape", spidermanEscape);  
   
 Attraction iceAgeAdventure = new Attraction();  
 iceAgeAdventure.setAttractType("Thrill Rides");  
 iceAgeAdventure.setAttractName("Ice Age Adventure");  
 iceAgeAdventure.setAge(">=8");  
 iceAgeAdventure.setHeight("<=200");  
 *attracMap*.put("Ice Age Adventure", iceAgeAdventure);  
   
 Attraction canyonBlaster = new Attraction();  
 canyonBlaster.setAttractType("Thrill Rides");  
 canyonBlaster.setAttractName("Canyon Blaster");  
 canyonBlaster.setAge(">=8");  
 canyonBlaster.setHeight(">=120");  
 *attracMap*.put("Canyon Blaster", canyonBlaster);  
   
 Attraction Theatre = new Attraction();  
 Theatre.setAttractType("Family Fun");  
 Theatre.setAttractName("4D Theatre");  
 Theatre.setAge("none");  
 Theatre.setHeight("none");  
 *attracMap*.put("4D Theatre", Theatre);  
   
 Attraction flowRider = new Attraction();  
 flowRider.setAttractType("Family Fun");  
 flowRider.setAttractName("Flow Rider");  
 flowRider.setAge("none");  
 flowRider.setHeight(">=100");  
 *attracMap*.put("Flow Rider", flowRider);  
   
 Attraction carousel = new Attraction();  
 carousel.setAttractType("Family Fun");  
 carousel.setAttractName("Carousel");  
 carousel.setAge("none");  
 carousel.setHeight("<=100");  
 *attracMap*.put("Carousel", carousel);  
 }  
   
}

1. *Attration.java*

package Entity;  
  
*/\*\*  
 \* Entity class for Attractions  
 \** ***@version*** *1.0  
 \*  
 \*/*public class Attraction {  
   
 private String attractType;  
 private String attractName;  
 private String age;  
 private String height;  
   
 public String getAttractType() {  
 return attractType;  
 }  
 public void setAttractType(String attractType) {  
 this.attractType = attractType;  
 }  
 public String getAttractName() {  
 return attractName;  
 }  
 public void setAttractName(String attractName) {  
 this.attractName = attractName;  
 }  
 public String getAge() {  
 return age;  
 }  
 public void setAge(String age) {  
 this.age = age;  
 }  
 public String getHeight() {  
 return height;  
 }  
 public void setHeight(String height) {  
 this.height = height;  
 }  
   
}

1. *InstructionService.java*

package Service;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.List;  
import java.util.Map;  
import java.util.Map.Entry;  
  
import Entity.Attraction;  
import Entity.Card;  
import Entity.Constants;  
  
*/\*\*  
 \* Class for operations of Instructions  
 \*   
 \** ***@version*** *2.0  
 \*   
 \*/*public class InstructionService {  
  
 private InstructionHelper helper = new InstructionHelper();  
   
 public void add(String instruction, HashMap<String, Card> cardMap) {  
 instruction = instruction.substring(4, instruction.length()).trim();  
 String[] addInfo = instruction.split(";");  
// String newId = instruction.substring(3, 9); // fatal error  
  
 String newId = addInfo[0].substring(3); // fatal correction!!!!! 1019 lab  
 if (cardMap.containsKey(newId)) {// update record  
  
 Card card = cardMap.get(newId);  
 helper.updateOrInsertCard(card, addInfo);  
  
 } else {// add new record  
 Card card = new Card();  
 helper.updateOrInsertCard(card, addInfo);  
  
 cardMap.put(newId, card);  
 }  
 }  
  
  
  
 public void delete(String instruction, HashMap<String, Card> cardMap) {  
  
 String delId = instruction.substring(10, instruction.length());  
 if (cardMap.containsKey(delId)) {// delete record  
 cardMap.remove(delId);  
  
 } else {// id does not exist  
 System.*out*.println("ERROR!Can not delete ID " + delId  
 + " because it does not exist!");  
 }  
 }  
  
   
  
   
  
 public void request(String instruction, HashMap<String, Card> cardMap, String filePath) {  
 instruction = instruction.substring(8, instruction.length());  
 helper.judgeRequest(instruction, cardMap, filePath);  
  
 }  
  
 // query results and write results to file  
 public void query(String queryStr, HashMap<String, Card> cardMap,  
 String filePath) {  
 if (queryStr.contains("name")) {  
 String name = queryStr.substring(11, queryStr.length());  
 helper.queryByName(name, cardMap, filePath);  
   
 } else if (queryStr.contains("ID")) {  
  
 String[] queryInfo = queryStr.substring(6, queryStr.length())  
 .split(";");  
 queryInfo[0] = queryInfo[0].trim();  
 queryInfo[1] = queryInfo[1].trim();  
 queryInfo[2] = queryInfo[2].trim().substring(3);  
  
 try {  
 helper.queryByID(queryInfo, cardMap, filePath);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 } else if (queryStr.contains("age")) {  
 String[] queryInfo = queryStr.substring(6, queryStr.length()).split(";");  
 queryInfo[0] = queryInfo[0].trim();  
 queryInfo[1] = queryInfo[1].trim();  
 helper.queryByAge(queryInfo, cardMap, filePath);  
 }  
 }  
  
   
  
 public void readInstrcutionFile(String instructFilePath,  
 HashMap<String, Card> cardMap, String reportFilePath) {  
 try {  
 String encoding = "utf-8";  
 File file = new File(instructFilePath);  
  
 if (file.isFile() && file.exists()) { // make a judgement about if file exists  
  
 InputStreamReader read = new InputStreamReader(  
 new FileInputStream(file), encoding);  
 BufferedReader bufferedReader = new BufferedReader(read);  
 String lineTxt = null;  
  
 while ((lineTxt = bufferedReader.readLine()) != null) {  
  
 if (lineTxt.startsWith("add ")) {  
 add(lineTxt.trim(), cardMap);  
 } else if (lineTxt.startsWith("delete ")) {  
 delete(lineTxt.trim(), cardMap);  
 } else if (lineTxt.startsWith("request ")) {  
 request(lineTxt.trim(), cardMap, reportFilePath);  
  
 } else if (lineTxt.startsWith("query ")) {  
 query(lineTxt.trim(), cardMap, reportFilePath);  
  
 }  
 }  
 read.close();  
 } else {  
 System.*out*.println("ERROR!Can not find specified file.");  
 }  
 } catch (Exception e) {  
 System.*out*.println("ERROR!Error occurs when reading files");  
 e.printStackTrace();  
 }  
 }  
  
   
}

1. *InstructionHelper.java*

package Service;  
  
import java.io.FileWriter;  
import java.io.IOException;  
import java.security.spec.ECField;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Comparator;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.List;  
import java.util.Map;  
import java.util.Map.Entry;  
  
import Entity.Attraction;  
import Entity.Card;  
import Entity.Constants;  
  
public class InstructionHelper {  
  
 public SimpleDateFormat toDateForamt(String biDay) {  
  
 SimpleDateFormat sdf = null;  
  
 if (biDay.contains("-")) {  
 sdf = new SimpleDateFormat("dd-MM-yyyy");  
  
 } else if (biDay.contains("/")) {  
 sdf = new SimpleDateFormat("dd/MM/yyyy");  
 }  
  
 return sdf;  
 }  
  
 public int calcuAge(String birthday) {  
  
// SimpleDateFormat sdf = null;  
//  
// if (birthday.contains("-")) {  
// sdf = new SimpleDateFormat("dd-MM-yyyy");  
//  
// } else if (birthday.contains("/")) {  
// sdf = new SimpleDateFormat("dd/MM/yyyy");  
// }  
  
 SimpleDateFormat sdf = null;  
 sdf = toDateForamt(birthday);  
  
// if (birthday.isEmpty() || birthday == null) {  
// System.out.println("================================================================");  
// } else {  
// System.out.println("========================" + birthday + "========================");  
// }  
  
 Date now = new Date();  
 int age = 0;  
 try {  
 Date birthDate = toDateForamt(birthday).parse(birthday);  
  
 long nowTime = now.getTime();  
 long birthTime = birthDate.getTime();  
 long interval = Math.*abs*(nowTime - birthTime);  
 age = (int) (interval / 1000 / 60 / 60 / 24 / 365);  
  
 } catch (ParseException e) {  
 System.*out*.println("birthday not valid");  
 }  
  
 return age;  
 }  
   
 public void updateOrInsertCard(Card card, String[] addInfo) { // add  
 for (int i = 0; i < addInfo.length; i++) {  
  
 if (addInfo[i].trim().contains("ID")) {  
 card.setId(addInfo[i].trim().substring(3, addInfo[i].trim().length())); // second arg: 9 or addInfo[i].trim().length() ?  
 } else if (addInfo[i].trim().contains("name")) {  
 card.setName(addInfo[i].trim().substring(5,  
 addInfo[i].trim().length()));  
 } else if (addInfo[i].trim().contains("birthday")) {  
 card.setBirthday(addInfo[i].trim().substring(9,  
 addInfo[i].trim().length()));  
 } else if (addInfo[i].trim().contains("height")) {  
 card.setHeight(addInfo[i].trim().substring(7,  
 addInfo[i].trim().length()));  
 } else if (addInfo[i].trim().contains("address")) {  
 card.setAddress(addInfo[i].trim().substring(8,  
 addInfo[i].trim().length()));  
 }  
 }  
 }  
   
 public void judgeRequest(String instruction, HashMap<String, Card> cardMap, String filePath) {  
 String content = "";  
 String[] requestInfo = instruction.split(";");  
 String requestId = requestInfo[0].substring(3, requestInfo[0].length());  
  
 if (cardMap.containsKey(requestId)) {// request  
 Card card = cardMap.get(requestId);  
  
 String birthday = card.getBirthday();  
 int age = calcuAge(birthday);  
  
 try {  
 int height = Integer.*valueOf*(card.getHeight().substring(0, // error  
 card.getHeight().length() - 2));  
  
 String attracName = requestInfo[1].trim();  
 Attraction attrac = Constants.*attracMap*.get(attracName);  
 String ageRequire = attrac.getAge();  
 String heightRequire = attrac.getHeight();  
 boolean flag = true;  
  
 if (ageRequire.contains(">=")) {  
 if (age < Integer.*valueOf*(ageRequire.substring(2,  
 ageRequire.length()))) {  
// System.out.println("----request " + instruction + "---");  
// System.out.println("Request Denied: " + requestInfo[1]  
// + " " + requestInfo[2]);  
// System.out.println("Reasons: Age requirement not met");  
 flag = false;  
  
 content = "----request " + instruction + "---" + "\r\n" +  
 "Request Denied: " + requestInfo[1] + " " + requestInfo[2] + "\r\n" +  
 "Reasons: Age requirement not met";  
  
 System.*out*.println(content);  
  
 content += "\r\n---------------------------------------\r\n\n";  
  
 }  
 } else if (ageRequire.contains("<=")) {  
 if (age > Integer.*valueOf*(ageRequire.substring(2,  
 ageRequire.length()))) {  
// System.out.println("----request " + instruction + "---");  
// System.out.println("Request Denied: " + requestInfo[1]  
// + " " + requestInfo[2]);  
// System.out.println("Reasons: Age requirement not met");  
 flag = false;  
  
 content = "----request " + instruction + "---" + "\r\n" +  
 "Request Denied: " + requestInfo[1] + " " + requestInfo[2] + "\r\n" +  
 "Reasons: Age requirement not met";  
  
 System.*out*.println(content);  
  
 content += "\r\n---------------------------------------\r\n\n";  
  
 }  
 }  
  
 if (heightRequire.contains(">=")) {  
 if (height < Integer.*valueOf*(heightRequire.substring(2,  
 heightRequire.length()))) {  
// System.out.println("----request " + instruction + "---");  
// System.out.println("Request Denied: " + requestInfo[1]  
// + " " + requestInfo[2]);  
// System.out.println("Reasons: Height requirement not met");  
 flag = false;  
  
 content = "----request " + instruction + "---" + "\r\n" +  
 "Request Denied: " + requestInfo[1] + " " + requestInfo[2] + "\r\n" +  
 "Reasons: Height requirement not met";  
  
 System.*out*.println(content);  
  
 content += "\r\n---------------------------------------\r\n\n";  
  
 }  
 } else if (heightRequire.contains("<=")) {  
 if (height > Integer.*valueOf*(heightRequire.substring(2,  
 heightRequire.length()))) {  
// System.out.println("----request " + instruction + "---");  
// System.out.println("Request Denied: " + requestInfo[1]  
// + " " + requestInfo[2]);  
// System.out.println("Reasons: Height requirement not met");  
 flag = false;  
  
 content = "----request " + instruction + "---" + "\r\n" +  
 "Request Denied: " + requestInfo[1] + " " + requestInfo[2]+ "\r\n" +  
 "Reasons: Height requirement not met";  
  
 System.*out*.println(content);  
  
 content += "\r\n---------------------------------------\r\n\n";  
 }  
 }  
  
 if (flag) {  
 String visitHistory = card.getAttracVisitHistory();  
 card.setAttracVisitHistory(visitHistory + "\n" + requestInfo[1]  
 + " " + requestInfo[2]);  
  
 }  
 } catch (Exception e) {  
 e.getMessage();  
 }  
  
 } else {// id does not exist  
// System.out.println("----request " + instruction + "---");  
// System.out.println("Request Denied: " + requestInfo[1] + " "  
// + requestInfo[2]);  
// System.out.println("Reasons: Request ID does not exist");  
  
 content = "----request " + instruction + "---" + "\r\n" +  
 "Request Denied: " + requestInfo[1] + " " + requestInfo[2] + "\r\n" +  
 "Reasons: Request ID does not exist";  
  
 System.*out*.println(content);  
 content += "\r\n---------------------------------------\r\n\n";  
  
 }  
  
 if (content != "") {  
 *appendContent*(filePath, content);  
 }  
 }  
   
   
 public void queryByName(String name, HashMap<String, Card> cardMap,  
 String filePath) {  
  
 Iterator iter = cardMap.entrySet().iterator();  
 String content = "";  
 while (iter.hasNext()) {  
 Entry entry = (Map.Entry) iter.next();  
 String id = (String) entry.getKey();  
 Card card = (Card) entry.getValue();  
 if (name.equals(card.getName())) {  
 content += "----query name " + name + "----\r\n";  
 if (card.getAttracVisitHistory() != null) {  
 String[] attracHist = card.getAttracVisitHistory().split("#");  
 for (int i = 0; i < attracHist.length; i++) {  
 content += attracHist[i] + "\r\n";  
 }  
 }  
 content += "---------------------------------------\r\n";  
 }  
 }  
 *appendContent*(filePath, content);  
 }  
  
 public void queryByID(String[] queryInfo, HashMap<String, Card> cardMap,  
 String filePath) throws ParseException {  
  
 SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");  
 HashMap<String, Integer> visitMap = new HashMap<String, Integer>();  
  
 Iterator iter = cardMap.entrySet().iterator();  
 while (iter.hasNext()) {  
 Entry entry = (Map.Entry) iter.next();  
 String id = (String) entry.getKey();  
 Card card = (Card) entry.getValue();  
 if (queryInfo[2].equals(card.getId())) {  
 if (card.getAttracVisitHistory().contains("-")) {  
 card.setAttracVisitHistory(card.getAttracVisitHistory()  
 .replace("-", "/"));  
 }  
 Date fromDate = null;  
 Date toDate = null;  
 try {  
 fromDate = toDateForamt(queryInfo[0]).parse(queryInfo[0]);  
 toDate = toDateForamt(queryInfo[1]).parse(queryInfo[1]);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
  
 String[] visitHistory = card.getAttracVisitHistory().split("#");  
 int num = visitHistory.length;  
  
 for (int i = 0; i < num; i++) {  
 if (visitHistory[i].startsWith("4D Theatre")) {  
 visitMap.put(  
 "4D Theatre",  
 getIndex("4D Theatre", visitHistory[i],  
 fromDate, toDate, 11));  
 } else if (visitHistory[i].startsWith("Spiderman Escape")) {  
 visitMap.put(  
 "Spiderman Escape",  
 getIndex("Spiderman Escape", visitHistory[i],  
 fromDate, toDate, 16));  
 } else if (visitHistory[i].startsWith("Ice Age Adventure")) {  
 visitMap.put(  
 "Ice Age Adventure",  
 getIndex("Ice Age Adventure", visitHistory[i],  
 fromDate, toDate, 16));  
 } else if (visitHistory[i].startsWith("Canyon Blaster")) {  
 visitMap.put(  
 "Canyon Blaster",  
 getIndex("Canyon Blaster", visitHistory[i],  
 fromDate, toDate, 14));  
 } else if (visitHistory[i].startsWith("Flow Rider")) {  
 visitMap.put(  
 "Flow Rider",  
 getIndex("Flow Rider", visitHistory[i],  
 fromDate, toDate, 11));  
 } else if (visitHistory[i].startsWith("Carousel")) {  
 visitMap.put(  
 "Carousele",  
 getIndex("Carousel", visitHistory[i], fromDate,  
 toDate, 9));  
 }  
 }  
  
 Iterator iterHist = visitMap.entrySet().iterator();  
 int totalVisits = 0;  
 int mostVisits = 0;  
 int secondVisits = 0;  
  
 while (iterHist.hasNext()) {  
 Entry entryHist = (Map.Entry) iterHist.next();  
 String attracName = (String) entryHist.getKey();  
 int index = (int) entryHist.getValue();  
 totalVisits += index;  
 }  
  
 if (totalVisits != 0) {  
 String content = "";  
 content += "----query " + queryInfo[0] + "; "  
 + queryInfo[1] + "; ID " + queryInfo[2]  
 + "----\r\n";  
 content += "Total visits: " + totalVisits + "\r\n";  
  
 List<Map.Entry<String, Integer>> list = new ArrayList<Map.Entry<String, Integer>>(  
 visitMap.entrySet());  
 Collections.*sort*(list,  
 new Comparator<Map.Entry<String, Integer>>() {  
 // Order  
 @Override  
 public int compare(Entry<String, Integer> o1,  
 Entry<String, Integer> o2) {  
 return o2.getValue().compareTo(  
 o1.getValue());  
 }  
 });  
 int output = 0;  
  
 for (Map.Entry<String, Integer> mapping : list) {  
 if (output == 0) {  
 content += "Most-visited: " + mapping.getKey()  
 + " " + mapping.getValue() + "\r\n";  
 output++;  
 } else if (output == 1) {  
 content += "2nd-most-visited: " + mapping.getKey()  
 + " " + mapping.getValue() + "\r\n";  
 output++;  
 }  
 }  
  
 content += "---------------------------------------\r\n";  
 *appendContent*(filePath, content);  
 }  
 }  
 }  
 }  
  
 public int getIndex(String attracName, String visitHistory, Date fromDate,  
 Date toDate, int offset) {  
 int index = 0;  
 SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");  
 String visitDate[] = visitHistory.substring(offset,  
 visitHistory.length()).split(" ");  
 for (int j = 0; j < visitDate.length; j++) {  
 Date visitDay = null;  
 if (visitDate[j].contains("/")) {  
 try {  
 visitDay = sdf.parse(visitDate[j]);  
 if (visitDay.after(fromDate) && visitDay.before(toDate)) {  
 index++;  
 }  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 return index;  
 }  
  
 public void queryByAge(String[] queryInfo, HashMap<String, Card> cardMap,  
 String filePath) {  
 Iterator iter = cardMap.entrySet().iterator();  
 int population = 0;  
 SimpleDateFormat sdf = null;  
 ArrayList<Integer> ageList = new ArrayList<Integer>();  
  
 while (iter.hasNext()) {  
 Entry entry = (Map.Entry) iter.next();  
 Card card = (Card) entry.getValue();  
 // Calculate age  
 int age = calcuAge(card.getBirthday());  
 if (card.getAttracVisitHistory() != null) {  
 if (card.getAttracVisitHistory().contains("-")) {  
 card.setAttracVisitHistory(card.getAttracVisitHistory().replace("-", "/"));  
 }  
 Date fromDate = null;  
 Date toDate = null;  
 try {  
// fromDate = sdf.parse(queryInfo[0]);  
// toDate = sdf.parse(queryInfo[1]);  
  
 fromDate = toDateForamt(queryInfo[0]).parse(queryInfo[0]);  
 toDate = toDateForamt(queryInfo[1]).parse(queryInfo[1]);  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
  
 ArrayList<String> visitDateList = new ArrayList<String>();  
 String[] visitHistory = card.getAttracVisitHistory().split("#");  
 int num = visitHistory.length;  
 for (int i = 0; i < num; i++) {  
 String[] histSegment = visitHistory[i].split(" ");  
 for (int j = 0; j < histSegment.length; j++) {  
 if (histSegment[j].contains("/") || histSegment[j].contains("-")) {  
 Date visitDay = null;  
 try {  
 visitDay = toDateForamt(histSegment[j]).parse(histSegment[j]);  
 if (visitDay.after(fromDate)&& visitDay.before(toDate)) {  
 ageList.add(age);  
 population++;  
 }  
 } catch (ParseException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 }  
 }  
 }  
 int below8 = 0;  
 int over8AndBelow18 = 0;  
 int over18AndBelow65 = 0;  
 int over65 = 0;  
  
 for (Iterator iterAgeList = ageList.iterator(); iterAgeList.hasNext();) {  
  
 int age = (Integer) iterAgeList.next();  
 if (age <= 8) {  
 below8++;  
 } else if (age > 8 && age <= 18) {  
 over8AndBelow18++;  
 } else if (age > 18 && age <= 65) {  
 over18AndBelow65++;  
 } else {  
 over65++;  
 }  
 }  
  
 String content = "";  
 if (population > 0) {  
 content += "----query " + queryInfo[0] + "; " + queryInfo[1]  
 + "; age----" + "\r\nPopulation size: " + population  
 + "\r\n" + "Age profile\r\nBelow 8: "  
 + ((float) below8 / population) \* 100  
 + "%\r\nOver 8 and below 18: "  
 + ((float) over8AndBelow18 / population) \* 100  
 + "%\r\nOver 18 and Below 65: "  
 + ((float) over18AndBelow65 / population) \* 100  
 + "%\r\nOver 65: " + ((float) over65 / population) \* 100  
 + "%\r\n";  
 content += "---------------------------------------\r\n";  
 *appendContent*(filePath, content);  
  
 }  
  
 }  
   
 public static void appendContent(String filePath, String content) {  
 try {  
 // Open a file writer, and in mode of appending  
 FileWriter writer = new FileWriter(filePath, true);  
 writer.write(content);  
 writer.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
   
   
}

1. *CardService.java*

package Service;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.Map;  
import java.util.Scanner;  
import java.util.Map.Entry;  
  
import Entity.Card;  
  
*/\*\*  
 \* Class for operations of Card  
 \*   
 \** ***@version*** *1.2  
 \*   
 \*/*public class CardService {  
  
 public HashMap<String, Card> readCardFile(String filePath) {  
  
 HashMap<String, Card> cardMap = new HashMap<String, Card>();  
  
 try {  
 File file = new File(filePath);  
  
 if (file.isFile() && file.exists()) { // make a judgement about if file exists  
  
 Scanner sc = null;  
 try {  
 sc = new Scanner(new FileReader(filePath));  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 // initialize variables  
 String lineTxt = null;  
 String address = "";  
 String attracHistory = "";  
 String id = "";  
 String name = "";  
 String birthday = "";  
 String height = "";  
  
 while ((sc.hasNextLine() && (lineTxt = sc.nextLine()) != null)) {  
  
 if (lineTxt.contains("ID")) {  
 id = lineTxt.substring(3, lineTxt.length());  
   
 } else if (lineTxt.contains("name")) {  
 name = lineTxt.substring(5, lineTxt.length());  
   
 } else if (lineTxt.contains("birthday")) {  
 birthday = lineTxt.substring(9, lineTxt.length());  
   
 } else if (lineTxt.contains("height")) {  
 height = lineTxt.substring(7, lineTxt.length());  
   
 } else if (lineTxt.contains("Spiderman Escape")  
 || lineTxt.contains("Ice Age Adventure")  
 || lineTxt.contains("Canyon Blaster")  
 || lineTxt.contains("4D Theatre")  
 || lineTxt.contains("Flow Rider")  
 || lineTxt.contains("Carousel")) {  
 attracHistory += lineTxt + "#"; // may error  
  
 } else {  
 if (lineTxt.contains("address")) {  
 lineTxt = lineTxt.substring(8, lineTxt.length());  
   
 }  
 address += lineTxt; // may error  
  
 }  
  
 if (attracHistory.length() != 0) {  
 }  
  
 if (lineTxt.length() == 0 ||!sc.hasNextLine()) {  
   
 Card card = new Card();  
 card.setId(id.trim());  
 card.setName(name.trim());  
 card.setHeight(height.trim());  
 card.setAddress(address);  
 card.setBirthday(birthday.trim());  
 card.setAttracVisitHistory(attracHistory);  
  
 cardMap.put(card.getId(), card);  
  
 address = "";  
 attracHistory = "";  
 }  
  
 }  
 }  
 } catch (Exception e) {  
 System.*out*.println("ERROR! Error occurs when reading files");  
 e.printStackTrace();  
 }  
 return cardMap;  
 }  
  
 public void appendContent(String filePath, String content) {  
 try {  
 // Open a file writer, but not in mode of appending  
 FileWriter writer = new FileWriter(filePath, false);  
 writer.write(content);  
 writer.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void writeCardIntoResultFile(String filePath, HashMap<String, Card> cardMap) {  
  
 Iterator iter = cardMap.entrySet().iterator();  
 String content = "";  
 while (iter.hasNext()) {  
 Entry entry = (Map.Entry) iter.next();  
 String id = (String) entry.getKey();  
 Card card = (Card) entry.getValue();  
 content += "ID " + id + "\r\n";  
 content += "name " + card.getName() + "\r\n";  
 content += "birthday " + card.getBirthday() + "\r\n";  
 if (card.getAddress() != null) {  
 content += "address " + card.getAddress() + "\r\n";  
 }  
 if (card.getHeight() != null) {  
 content += "height " + card.getHeight() + "\r\n";  
 }  
 if (card.getAttracVisitHistory() != null) {  
 String[] attracHistory = card.getAttracVisitHistory()  
 .split("#");  
 for (int k = 0; k < attracHistory.length; k++) {  
 content += attracHistory[k] + "\r\n";  
 }  
 }  
 content += "\r\n";  
 }  
  
 appendContent(filePath, content);  
 }  
}

1. *Card.java*

package Entity;  
  
*/\*\*  
 \* Entity class of Card  
 \** ***@version*** *1.0  
 \*  
 \*/*public class Card {  
   
 private String id;  
 private String name;  
 private String birthday;  
 private String height;  
 private String address;  
 private String attracVisitHistory;  
  
 public String getId() {  
 return id;  
 }  
  
 public void setId(String id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(String birthday) {  
 this.birthday = birthday;  
 }  
  
 public String getHeight() {  
 return height;  
 }  
  
 public void setHeight(String height) {  
 this.height = height;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public String getAttracVisitHistory() {  
 return attracVisitHistory;  
 }  
  
 public void setAttracVisitHistory(String attracVisitHistory) {  
 this.attracVisitHistory = attracVisitHistory;  
 }  
  
 public boolean isInfoValid() {  
 if (id != null && id != "" &&  
 name != null && name != "" &&  
 birthday != null && birthday != "" &&  
 height != null && height != "" &&  
 address != null && address != "") {  
 return true;  
 } else {  
 return false;  
 }  
 }  
  
}