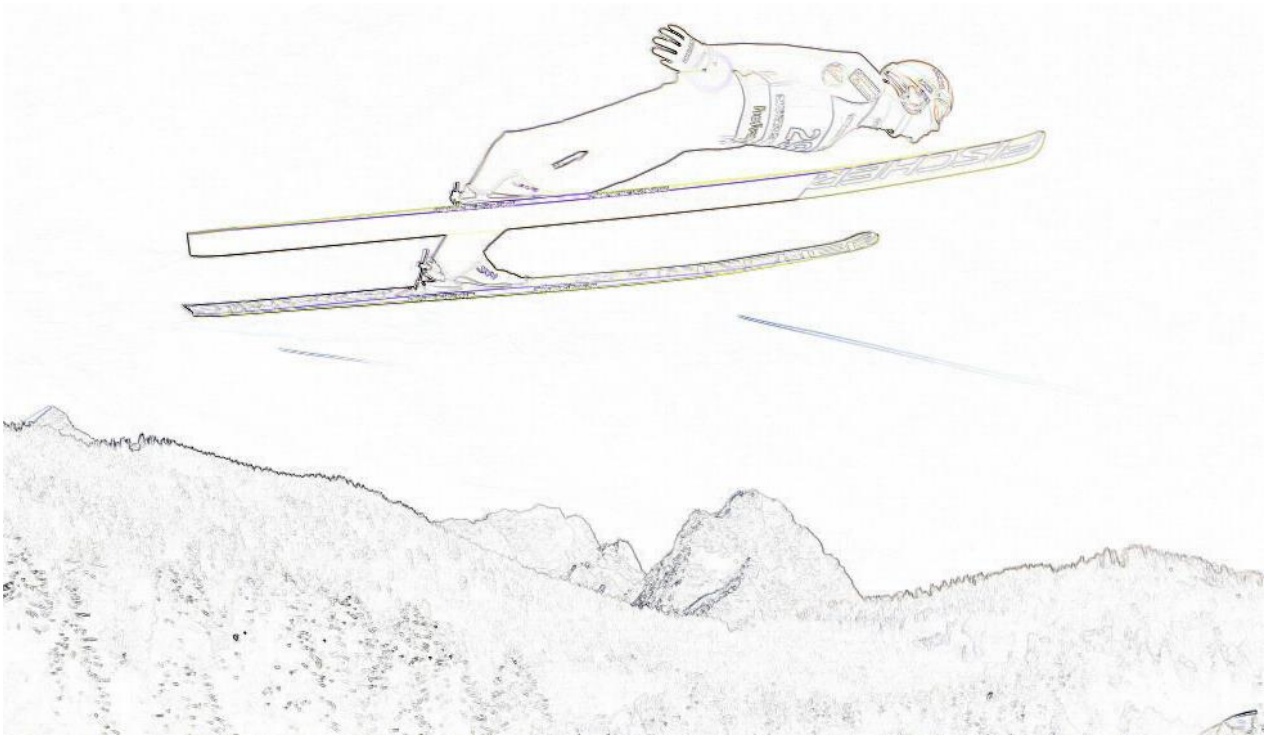
Ski Jumping Hill Records Database implementation document

Pawel Paszki (Applied Computing Year 2)

Student number: 20063617

Database Design



Contents

I. Gathering / generating data

I have gathered the real data for Hills, Countries, Coaches and Ski_jumpers tables from the internet (I did not found any data set, therefore I needed to get all the data myself, ie all dates of birth, hill addresses, their sizes, etc).

I would have been impossible (timewise) to gather all the hill records per ski jumper as well, so I have decided to create Java application, which will generate data (which is not that far out from the real data) for the longest jumps. I needed to hardcode all ski jumpers' details (like dob, id; and subjectively rank them from 6-10, so that the generated hill records will correspond with the actual level of the ski jumpers), hill details, competitions' dates (>2000yr) and actual records to make sure that generated lengths of jumps will be lower than the actual records. The records themselves (actual hills' records) needed to be updated after the generated data was populated into the table. Note – ski jumpers with lower ranks and short starting history deliberately don't have their hill records in all the hills, in which there were WC competitions held.

Here is the code for the generator:

Hill class:

```
import java.util.Calendar;

public class Hill {
	private int hillSize;
	private int kPoint;
	private double record;
	private Calendar[] dates;
	private Calendar latestDate;
	public Hill(int hillSize, int kPoint, double record) {
		this.hillSize = hillSize;
		this.kPoint = kPoint;
		this.record = record;
	}
	public int getHillSize() {
		return hillSize;
	}
	public void setHillSize(int hillSize) {
		this.hillSize = hillSize;
	}
	public double getRecord() {
		return record;
	}
	public void setRecord(double record) {
		this.record = record;
	}
	public int getkPoint() {
		return kPoint;
	}
	public void setkPoint(int kPoint) {
		this.kPoint = kPoint;
	}
	public Calendar[] getDates() {
		return dates;
	}
	public void setDates(Calendar[] dates) {
		this.dates = dates;
	}
	public Calendar getLatestDate() {
		return latestDate;
	}
	public void setLatestDate(Calendar latestDate) {
		this.latestDate = latestDate;
	}
}
```

Ski_Jumper class:

```
import java.util.Calendar;
import java.util.GregorianCalendar;

public class Ski_Jumper {
	private int id;
	private int rank;
	private Calendar dateOfBirth;
	private Calendar startDate;
	private Calendar finishDate;
	public Ski_Jumper(int id, int rank, int birthDay, int birthMonth, int birthYear, int startDay,
	int startMonth, int startYear) {
		this.setId(id);
		this.rank = rank;
		dateOfBirth = new GregorianCalendar(birthYear, birthMonth, birthDay);
		startDate = new GregorianCalendar(startYear, startMonth, startDay);
	}
	public Ski_Jumper(int id, int rank, int birthDay, int birthMonth, int birthYear, int startDay,
	int startMonth, int startYear, int finishDay, int finishMonth, int finishYear) {
		this.setId(id);
		this.rank = rank;
		this.dateOfBirth = new GregorianCalendar(birthYear, birthMonth, birthDay);
		startDate = new GregorianCalendar(startYear, startMonth, startDay);
		finishDate = new GregorianCalendar(finishYear, finishMonth, finishDay);
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public int getRank() {
		return rank;
	}
	public void setRank(int rank) {
		this.rank = rank;
	}
	public Calendar getDateOfBirth() {
		return dateOfBirth;
	}
	public void setDateOfBirth(Calendar dateOfBirth) {
		this.dateOfBirth = dateOfBirth;
	}
	public Calendar getStartDate() {
		return startDate;
	}
	public void setStartDate(Calendar startDate) {
		this.startDate = startDate;
	}
	public Calendar getFinishDate() {
		return finishDate;
	}
	public void setFinishDate(Calendar finishDate) {
		this.finishDate = finishDate;
	}
}
```

Generator class:

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Random;
import java.util.TreeSet;

public class Generator {
	private Ski_Jumper[] skiJumpers = new Ski_Jumper[97];
```

```
private Hill[] hills = new Hill[24];
private Random random = new Random();
private ArrayList<Integer> unattended;
private TreeSet<Integer> hillsNum;
private int id;
public static void main(String... args) {
        Generator generator = new Generator();
        generator.instantiate();
}
private void instantiate() {
        skiJumpers[0] = new Ski_Jumper(1, 10, 11, 4, 1988, 22, 11, 2007);
        skiJumpers[1] = new Ski_Jumper(2, 10, 20, 8, 1992, 05, 11, 2009);
        skiJumpers[2] = new Ski_Jumper(3, 10, 13, 4, 1993, 03, 0, 2008);
        skiJumpers[3] = new Ski_Jumper(4, 9, 13, 4, 1991, 13, 11, 2008);
        skiJumpers[4] = new Ski_Jumper(5, 10, 5, 2, 1991, 3, 2, 2008);
        skiJumpers[5] = new Ski_Jumper(6, 10, 6, 5, 1972, 20, 11, 1989);
        skiJumpers[6] = new Ski_Jumper(7, 9, 9, 6, 1989, 7, 1, 2004);
        skiJumpers[7] = new Ski_Jumper(8, 8, 19, 6, 1989, 7, 2, 2009);
        skiJumpers[8] = new Ski_Jumper(9, 10, 25, 4, 1987, 17, 0, 2004);
        skiJumpers[9] = new Ski_Jumper(10, 10, 7, 0, 1990, 12, 2, 2006);
        skiJumpers[10] = new Ski_Jumper(11, 10, 25, 5, 1981, 29, 11, 1997);
        skiJumpers[11] = new Ski_Jumper(12, 9, 14, 7, 1991, 17, 0, 2009);
        skiJumpers[12] = new Ski_Jumper(13, 9, 14, 1, 1989, 3, 0, 2003);
        skiJumpers[13] = new Ski_Jumper(14, 8, 28, 4, 1983, 1, 0, 2003);
        skiJumpers[14] = new Ski_Jumper(15, 7, 3, 3, 1991, 4, 0, 2009);
        skiJumpers[15] = new Ski_Jumper(16, 10, 27, 11, 1985, 13, 2, 2002);
        skiJumpers[16] = new Ski_Jumper(17, 10, 17, 1, 1985, 5, 11, 2004);
        skiJumpers[17] = new Ski_Jumper(18, 10, 24, 7, 1982, 3, 1, 2001);
        skiJumpers[18] = new Ski_Jumper(19, 8, 16, 0, 1987, 21, 0, 2006);
        skiJumpers[19] = new Ski_Jumper(20, 6, 20, 4, 1987, 24, 10, 2006);
        skiJumpers[20] = new Ski_Jumper(21, 6, 15, 0, 1979, 24, 0, 2002);
        skiJumpers[21] = new Ski_Jumper(22, 6, 13, 1, 1991, 17, 1, 2009);
        skiJumpers[22] = new Ski_Jumper(23, 7, 4, 6, 1995, 24, 0, 2013);
        skiJumpers[23] = new Ski_Jumper(24, 7, 7, 11, 1992, 9, 1, 2010);
        skiJumpers[24] = new Ski_Jumper(25, 7, 14, 7, 1990, 11, 2, 2008);
        skiJumpers[25] = new Ski_Jumper(26, 9, 16, 6, 1981, 13, 2, 1998);
        skiJumpers[26] = new Ski_Jumper(27, 7, 27, 1, 1991, 13, 0, 2009);
        skiJumpers[27] = new Ski_Jumper(28, 10, 17, 4, 1984, 11, 2, 2000);
        skiJumpers[28] = new Ski_Jumper(29, 7, 28, 4, 1989, 12, 1, 2006);
        skiJumpers[29] = new Ski_Jumper(30, 7, 17, 5, 1985, 12, 0, 2002);
        skiJumpers[30] = new Ski_Jumper(31, 8, 26, 11, 1979, 27, 10, 1999);
        skiJumpers[31] = new Ski_Jumper(32, 8, 28, 7, 1995, 27, 1, 2012);
        skiJumpers[32] = new Ski_Jumper(33, 7, 30, 6, 1989, 5, 2, 2003);
        skiJumpers[33] = new Ski_Jumper(34, 7, 19, 4, 1987, 4, 1, 2006);
        skiJumpers[34] = new Ski_Jumper(35, 9, 22, 8, 1987, 5, 11, 2005);
        skiJumpers[35] = new Ski_Jumper(36, 10, 29, 0, 1980, 7, 1, 1988);
        skiJumpers[36] = new Ski_Jumper(37, 10, 15, 4, 1989, 9, 2, 2008);
        skiJumpers[37] = new Ski_Jumper(38, 7, 23, 10, 1994, 21, 0, 2011);
        skiJumpers[38] = new Ski_Jumper(39, 9, 25, 1, 1992, 3, 0, 2011);
        skiJumpers[39] = new Ski_Jumper(40, 7, 11, 5, 1991, 11, 0, 2009);
        skiJumpers[40] = new Ski_Jumper(41, 7, 21, 11, 1989, 21, 0, 2006);
        skiJumpers[41] = new Ski_Jumper(42, 6, 31, 9, 1994, 21, 10, 2014);
        skiJumpers[42] = new Ski_Jumper(43, 7, 8, 2, 1994, 30, 10, 2011);
        skiJumpers[43] = new Ski_Jumper(44, 7, 3, 7, 1990, 16, 0, 2010);
        skiJumpers[44] = new Ski_Jumper(45, 8, 24, 5, 1991, 27, 10, 2011);
        skiJumpers[45] = new Ski_Jumper(46, 6, 18, 1, 1988, 19, 1, 2004);
        skiJumpers[46] = new Ski_Jumper(47, 7, 10, 8, 1988, 15, 11, 2007);
        skiJumpers[47] = new Ski_Jumper(48, 9, 11, 3, 1990, 1, 11, 2007);
        skiJumpers[48] = new Ski_Jumper(49, 7, 12, 2, 1990, 16, 0, 2009);
        skiJumpers[49] = new Ski_Jumper(50, 6, 1, 7, 1993, 8, 0, 2013);
        skiJumpers[50] = new Ski_Jumper(51, 10, 27, 3, 1978, 9, 2, 1997);
        skiJumpers[51] = new Ski_Jumper(52, 6, 28, 9, 1991, 3, 2, 2012);
        skiJumpers[52] = new Ski_Jumper(53, 10, 11, 4, 1977, 29, 11, 1993, 10, 2, 2011);
        skiJumpers[53] = new Ski_Jumper(54, 6, 12, 1, 1987, 10, 0, 2004);
        skiJumpers[54] = new Ski_Jumper(55, 6, 20, 3, 1996, 20, 2, 2014);
        skiJumpers[55] = new Ski_Jumper(56, 7, 16, 1, 1991, 1, 11, 2007);
        skiJumpers[56] = new Ski_Jumper(57, 8, 3, 6, 1988, 5, 2, 2004);
        skiJumpers[57] = new Ski_Jumper(58, 8, 9, 5, 1988, 13, 11, 2007);
        skiJumpers[58] = new Ski_Jumper(59, 6, 28, 2, 1995, 8, 1, 2013);
        skiJumpers[59] = new Ski_Jumper(60, 7, 14, 6, 1993, 13, 11, 2008);
        skiJumpers[60] = new Ski_Jumper(61, 9, 13, 0, 1980, 6, 1, 1997, 20, 2, 2008);
        skiJumpers[61] = new Ski_Jumper(62, 6, 18, 6, 1989, 9, 0, 2011);
        skiJumpers[62] = new Ski_Jumper(63, 7, 17, 7, 1986, 20, 11, 2003);
        skiJumpers[63] = new Ski_Jumper(64, 7, 23, 2, 1990, 28, 10, 2013);
```

```
skiJumpers[64] = new Ski_Jumper(65, 7, 11, 1, 1993, 23, 10, 2012);
skiJumpers[65] = new Ski_Jumper(66, 7, 10, 8, 1984, 11, 0, 2003);
skiJumpers[66] = new Ski_Jumper(67, 7, 29, 8, 1986, 29, 11, 2005);
skiJumpers[67] = new Ski_Jumper(68, 7, 2, 10, 1985, 6, 11, 2008);
skiJumpers[68] = new Ski_Jumper(69, 5, 3, 11, 1983, 14, 1, 2012);
skiJumpers[69] = new Ski_Jumper(70, 9, 15, 0, 1984, 29, 10, 2002);
skiJumpers[70] = new Ski_Jumper(71, 6, 20, 4, 1990, 12, 11, 2013);
skiJumpers[71] = new Ski_Jumper(72, 7, 2, 1, 1991, 24, 0, 2014);
skiJumpers[72] = new Ski_Jumper(73, 6, 13, 3, 1998, 2, 0, 2016);
skiJumpers[73] = new Ski_Jumper(74, 8, 21, 4, 1994, 16, 1, 2013);
skiJumpers[74] = new Ski_Jumper(75, 8, 15, 6, 1993, 19, 0, 2012);
skiJumpers[75] = new Ski_Jumper(76, 7, 8, 7, 1982, 23, 0, 1999);
skiJumpers[76] = new Ski_Jumper(77, 6, 29, 7, 1993, 22, 10, 2013);
skiJumpers[77] = new Ski_Jumper(78, 6, 7, 6, 1991, 24, 0, 2009);
skiJumpers[78] = new Ski_Jumper(79, 7, 14, 6, 1988, 17, 0, 2010);
skiJumpers[79] = new Ski_Jumper(80, 7, 15, 5, 1990, 14, 11, 2008);
skiJumpers[80] = new Ski_Jumper(81, 6, 30, 3, 1991, 28, 10, 2009);
skiJumpers[81] = new Ski_Jumper(82, 6, 10, 8, 1993, 23, 10, 2012);
skiJumpers[82] = new Ski_Jumper(83, 7, 29, 3, 1992, 23, 10, 2012);
skiJumpers[83] = new Ski_Jumper(84, 6, 2, 7, 1988, 11, 0, 2009);
skiJumpers[84] = new Ski_Jumper(85, 8, 12, 2, 1996, 2, 0, 2013);
skiJumpers[85] = new Ski_Jumper(86, 9, 22, 0, 1982, 4, 0, 1999, 20, 2, 2009);
skiJumpers[86] = new Ski_Jumper(87, 6, 6, 7, 1989, 1, 0, 2009);
skiJumpers[87] = new Ski_Jumper(88, 7, 3, 7, 1991, 17, 0, 2009);
skiJumpers[88] = new Ski_Jumper(89, 7, 9, 8, 1987, 10, 11, 2005);
skiJumpers[89] = new Ski_Jumper(90, 7, 18, 1, 1983, 21, 11, 2002);
skiJumpers[90] = new Ski_Jumper(91, 6, 24, 1, 1992, 15, 0, 2011);
skiJumpers[91] = new Ski_Jumper(92, 6, 30, 7, 1988, 18, 11, 2005);
skiJumpers[92] = new Ski_Jumper(93, 7, 6, 6, 1990, 29, 10, 2008);
skiJumpers[93] = new Ski_Jumper(94, 6, 22, 6, 1982, 28, 10, 1999);
skiJumpers[94] = new Ski_Jumper(95, 6, 9, 3, 1985, 29, 10, 2003);
skiJumpers[95] = new Ski_Jumper(96, 10, 3, 11, 1977, 30, 10, 1996, 19, 2, 2011);
skiJumpers[96] = new Ski_Jumper(97, 10, 24, 0, 1994, 6, 11, 2013);
hills[0] = new Hill(225, 200, 244);
hills[1] = new Hill(130, 120, 138);
hills[2] = new Hill(140, 125, 132);
hills[3] = new Hill(205, 185, 214.5);
hills[4] = new Hill(134, 120, 145.5);
hills[5] = new Hill(134, 120, 139);
hills[6] = new Hill(127, 120, 136);
hills[7] = new Hill(142, 120, 147);
hills[8] = new Hill(130, 116, 135.5);
hills[9] = new Hill(213, 185, 225.5);
hills[10] = new Hill(140, 125, 143.5);
hills[11] = new Hill(140, 125, 147);
hills[12] = new Hill(145, 130, 152);
hills[13] = new Hill(140, 125, 144);
hills[14] = new Hill(134, 120, 143.5);
hills[15] = new Hill(225, 200, 251.5);
hills[16] = new Hill(138, 123, 146);
hills[17] = new Hill(134, 120, 141);
hills[18] = new Hill(134, 120, 139);
hills[19] = new Hill(134, 120, 140.5);
hills[20] = new Hill(225, 200, 248.5);
hills[21] = new Hill(124, 115, 135.5);
hills[22] = new Hill(137, 125, 142);
hills[23] = new Hill(134, 120, 133.5);
Calendar[] dates = new GregorianCalendar[13];
dates[0] = new GregorianCalendar(2003,2,1);
dates[1] = new GregorianCalendar(2003,2,2);
dates[2] = new GregorianCalendar(2005,0,15);
dates[3] = new GregorianCalendar(2005,0,16);
dates[4] = new GregorianCalendar(2009,0,10);
dates[5] = new GregorianCalendar(2009,0,11);
dates[6] = new GregorianCalendar(2010,0,9);
dates[7] = new GregorianCalendar(2010,0,10);
dates[8] = new GregorianCalendar(2012,1,15);
dates[9] = new GregorianCalendar(2014,0,11);
dates[10] = new GregorianCalendar(2014,0,12);
dates[11] = new GregorianCalendar(2015,0,10);
dates[12] = new GregorianCalendar(2015,0,11);
hills[0].setDates(dates);
hills[0].setLatestDate(new GregorianCalendar(2015, 0, 11));
dates = new GregorianCalendar[16];
```

```
dates[0] = new GregorianCalendar(2001,0,4);
dates[1] = new GregorianCalendar(2002,0,4);
dates[2] = new GregorianCalendar(2003,0,4);
dates[3] = new GregorianCalendar(2004,0,4);
dates[4] = new GregorianCalendar(2005,0,4);
dates[5] = new GregorianCalendar(2006,0,4);
dates[6] = new GregorianCalendar(2007,0,4);
dates[7] = new GregorianCalendar(2008,0,4);
dates[8] = new GregorianCalendar(2009,0,4);
dates[9] = new GregorianCalendar(2010,0,4);
dates[10] = new GregorianCalendar(2011,0,4);
dates[11] = new GregorianCalendar(2012,0,4);
dates[12] = new GregorianCalendar(2013,0,4);
dates[13] = new GregorianCalendar(2014,0,4);
dates[14] = new GregorianCalendar(2015,0,4);
dates[15] = new GregorianCalendar(2016,0,4);
hills[1].setDates(dates);
hills[1].setLatestDate(new GregorianCalendar(2016,0,4));
dates = new GregorianCalendar[16];
dates[0] = new GregorianCalendar(2001,0,6);
dates[1] = new GregorianCalendar(2002,0,6);
dates[2] = new GregorianCalendar(2003,0,6);
dates[3] = new GregorianCalendar(2004,0,6);
dates[4] = new GregorianCalendar(2005,0,6);
dates[5] = new GregorianCalendar(2006,0,6);
dates[6] = new GregorianCalendar(2007,0,6);
dates[7] = new GregorianCalendar(2008,0,6);
dates[8] = new GregorianCalendar(2009,0,6);
dates[9] = new GregorianCalendar(2010,0,6);
dates[10] = new GregorianCalendar(2011,0,6);
dates[11] = new GregorianCalendar(2012,0,6);
dates[12] = new GregorianCalendar(2013,0,6);
dates[13] = new GregorianCalendar(2014,0,6);
dates[14] = new GregorianCalendar(2015,0,6);
dates[15] = new GregorianCalendar(2016,0,6);
hills[2].setDates(dates);
hills[2].setLatestDate(new GregorianCalendar(2016,0,6));
dates = new GregorianCalendar[6];
dates[0] = new GregorianCalendar(2001,0,13);
dates[1] = new GregorianCalendar(2008,0,20);
dates[2] = new GregorianCalendar(2011,0,8);
dates[3] = new GregorianCalendar(2011,0,9);
dates[4] = new GregorianCalendar(2013,1,3);
dates[5] = new GregorianCalendar(2014,2,14);
hills[3].setDates(dates);
hills[3].setLatestDate(new GregorianCalendar(2014,2,14));
dates = new GregorianCalendar[7];
dates[0] = new GregorianCalendar(2004,11,11);
dates[1] = new GregorianCalendar(2004,11,12);
dates[2] = new GregorianCalendar(2005,11,10);
dates[3] = new GregorianCalendar(2005,11,11);
dates[4] = new GregorianCalendar(2011,11,9);
dates[5] = new GregorianCalendar(2011,11,10);
dates[6] = new GregorianCalendar(2011,11,11);
hills[4].setDates(dates);
hills[4].setLatestDate(new GregorianCalendar(2011,11,11));
dates = new GregorianCalendar[5];
dates[0] = new GregorianCalendar(2003,0,11);
dates[1] = new GregorianCalendar(2004,0,10);
dates[2] = new GregorianCalendar(2004,0,11);
dates[3] = new GregorianCalendar(2008,1,8);
dates[4] = new GregorianCalendar(2008,1,9);
hills[5].setDates(dates);
hills[5].setLatestDate(new GregorianCalendar(2008,1,9));
dates = new GregorianCalendar[16];
dates[0] = new GregorianCalendar(2000,10,24);
dates[1] = new GregorianCalendar(2000,10,25);
dates[2] = new GregorianCalendar(2000,11,3);
dates[3] = new GregorianCalendar(2000,11,2);
dates[4] = new GregorianCalendar(2001,10,23);
dates[5] = new GregorianCalendar(2001,10,24);
dates[6] = new GregorianCalendar(2004,2,10);
dates[7] = new GregorianCalendar(2005,2,9);
dates[8] = new GregorianCalendar(2006,2,7);
```

```
dates[9] = new GregorianCalendar(2007,2,13);
dates[10] = new GregorianCalendar(2008,2,3);
dates[11] = new GregorianCalendar(2008,2,4);
dates[12] = new GregorianCalendar(2010,2,9);
dates[13] = new GregorianCalendar(2010,11,1);
dates[14] = new GregorianCalendar(2014,2,4);
dates[15] = new GregorianCalendar(2016,1,23);
hills[6].setDates(dates);
hills[6].setLatestDate(new GregorianCalendar(2016,1,23));
dates = new GregorianCalendar[22];
dates[0] = new GregorianCalendar(2002,10,29);
dates[1] = new GregorianCalendar(2002,10,30);
dates[2] = new GregorianCalendar(2003,10,28);
dates[3] = new GregorianCalendar(2003,10,30);
dates[4] = new GregorianCalendar(2004,10,27);
dates[5] = new GregorianCalendar(2004,10,28);
dates[6] = new GregorianCalendar(2005,10,26);
dates[7] = new GregorianCalendar(2006,10,24);
dates[8] = new GregorianCalendar(2007,11,1);
dates[9] = new GregorianCalendar(2007,10,30);
dates[10] = new GregorianCalendar(2008,10,29);
dates[11] = new GregorianCalendar(2009,10,28);
dates[12] = new GregorianCalendar(2009,10,27);
dates[13] = new GregorianCalendar(2010,10,28);
dates[14] = new GregorianCalendar(2010,10,27);
dates[15] = new GregorianCalendar(2011,10,27);
dates[16] = new GregorianCalendar(2012,11,1);
dates[17] = new GregorianCalendar(2012,10,30);
dates[18] = new GregorianCalendar(2013,10,29);
dates[19] = new GregorianCalendar(2013,10,30);
dates[20] = new GregorianCalendar(2014,10,28);
dates[21] = new GregorianCalendar(2014,10,29);
hills[7].setDates(dates);
hills[7].setLatestDate(new GregorianCalendar(2014,10,29));
dates = new GregorianCalendar[30];
dates[0] = new GregorianCalendar(2002,2,1);
dates[1] = new GregorianCalendar(2002,2,2);
dates[2] = new GregorianCalendar(2003,2,14);
dates[3] = new GregorianCalendar(2003,2,15);
dates[4] = new GregorianCalendar(2004,2,7);
dates[5] = new GregorianCalendar(2004,2,6);
dates[6] = new GregorianCalendar(2005,2,6);
dates[7] = new GregorianCalendar(2005,2,5);
dates[8] = new GregorianCalendar(2006,2,4);
dates[9] = new GregorianCalendar(2006,2,5);
dates[10] = new GregorianCalendar(2007,2,10);
dates[11] = new GregorianCalendar(2007,2,11);
dates[12] = new GregorianCalendar(2008,2,1);
dates[13] = new GregorianCalendar(2008,2,2);
dates[14] = new GregorianCalendar(2009,2,7);
dates[15] = new GregorianCalendar(2010,2,6);
dates[16] = new GregorianCalendar(2010,2,7);
dates[17] = new GregorianCalendar(2011,2,12);
dates[18] = new GregorianCalendar(2011,2,13);
dates[19] = new GregorianCalendar(2012,2,2);
dates[20] = new GregorianCalendar(2012,2,4);
dates[21] = new GregorianCalendar(2013,2,9);
dates[22] = new GregorianCalendar(2013,2,10);
dates[23] = new GregorianCalendar(2014,2,28);
dates[24] = new GregorianCalendar(2014,2,1);
dates[25] = new GregorianCalendar(2014,2,2);
dates[26] = new GregorianCalendar(2015,2,7);
dates[27] = new GregorianCalendar(2015,2,8);
dates[28] = new GregorianCalendar(2016,2,19);
dates[29] = new GregorianCalendar(2016,2,21);
hills[8].setDates(dates);
hills[8].setLatestDate(new GregorianCalendar(2016,2,21));
dates = new GregorianCalendar[15];
dates[0] = new GregorianCalendar(2001,2,3);
dates[1] = new GregorianCalendar(2001,2,4);
dates[2] = new GregorianCalendar(2004,1,7);
dates[3] = new GregorianCalendar(2007,0,27);
dates[4] = new GregorianCalendar(2007,0,28);
dates[5] = new GregorianCalendar(2009,1,14);
```

```
dates[6] = new GregorianCalendar(2009,1,15);
dates[7] = new GregorianCalendar(2010,0,30);
dates[8] = new GregorianCalendar(2010,0,31);
dates[9] = new GregorianCalendar(2011,1,5);
dates[10] = new GregorianCalendar(2011,1,6);
dates[11] = new GregorianCalendar(2012,1,18);
dates[12] = new GregorianCalendar(2012,1,19);
dates[13] = new GregorianCalendar(2013,1,16);
dates[14] = new GregorianCalendar(2013,1,17);
hills[9].setDates(dates);
hills[9].setLatestDate(new GregorianCalendar(2013,1,17));
dates = new GregorianCalendar[16];
dates[0] = new GregorianCalendar(2001,0,1);
dates[1] = new GregorianCalendar(2002,0,1);
dates[2] = new GregorianCalendar(2003,0,1);
dates[3] = new GregorianCalendar(2004,0,1);
dates[4] = new GregorianCalendar(2005,0,1);
dates[5] = new GregorianCalendar(2006,0,1);
dates[6] = new GregorianCalendar(2007,0,1);
dates[7] = new GregorianCalendar(2008,0,1);
dates[8] = new GregorianCalendar(2009,0,1);
dates[9] = new GregorianCalendar(2010,0,1);
dates[10] = new GregorianCalendar(2011,0,1);
dates[11] = new GregorianCalendar(2012,0,1);
dates[12] = new GregorianCalendar(2013,0,1);
dates[13] = new GregorianCalendar(2014,0,1);
dates[14] = new GregorianCalendar(2015,0,1);
dates[15] = new GregorianCalendar(2016,0,1);
hills[10].setDates(dates);
hills[10].setLatestDate(new GregorianCalendar(2016,0,1));
dates = new GregorianCalendar[12];
dates[0] = new GregorianCalendar(2007,1,7);
dates[1] = new GregorianCalendar(2009,1,1);
dates[2] = new GregorianCalendar(2010,1,3);
dates[3] = new GregorianCalendar(2011,1,2);
dates[4] = new GregorianCalendar(2012,1,15);
dates[5] = new GregorianCalendar(2013,1,13);
dates[6] = new GregorianCalendar(2013,10,23);
dates[7] = new GregorianCalendar(2013,10,24);
dates[8] = new GregorianCalendar(2014,10,22);
dates[9] = new GregorianCalendar(2014,10,23);
dates[10] = new GregorianCalendar(2015,10,21);
dates[11] = new GregorianCalendar(2015,10,22);
hills[11].setDates(dates);
hills[11].setLatestDate(new GregorianCalendar(2015,10,22));
dates = new GregorianCalendar[33];
dates[0] = new GregorianCalendar(2001,1,2);
dates[1] = new GregorianCalendar(2001,1,3);
dates[2] = new GregorianCalendar(2001,1,4);
dates[3] = new GregorianCalendar(2002,0,12);
dates[4] = new GregorianCalendar(2002,0,13);
dates[5] = new GregorianCalendar(2003,1,8);
dates[6] = new GregorianCalendar(2003,1,9);
dates[7] = new GregorianCalendar(2004,1,14);
dates[8] = new GregorianCalendar(2004,1,15);
dates[9] = new GregorianCalendar(2005,1,8);
dates[10] = new GregorianCalendar(2005,1,9);
dates[11] = new GregorianCalendar(2006,1,4);
dates[12] = new GregorianCalendar(2006,1,5);
dates[13] = new GregorianCalendar(2007,1,10);
dates[14] = new GregorianCalendar(2007,1,11);
dates[15] = new GregorianCalendar(2008,1,16);
dates[16] = new GregorianCalendar(2008,1,17);
dates[17] = new GregorianCalendar(2009,1,7);
dates[18] = new GregorianCalendar(2009,1,8);
dates[19] = new GregorianCalendar(2010,1,6);
dates[20] = new GregorianCalendar(2010,1,7);
dates[21] = new GregorianCalendar(2011,0,29);
dates[22] = new GregorianCalendar(2011,0,30);
dates[23] = new GregorianCalendar(2012,1,11);
dates[24] = new GregorianCalendar(2012,1,12);
dates[25] = new GregorianCalendar(2013,1,9);
dates[26] = new GregorianCalendar(2014,1,1);
dates[27] = new GregorianCalendar(2014,1,2);
```

```
dates[28] = new GregorianCalendar(2015,0,30);
dates[29] = new GregorianCalendar(2015,0,31);
dates[30] = new GregorianCalendar(2015,1,1);
dates[31] = new GregorianCalendar(2016,0,9);
dates[32] = new GregorianCalendar(2016,0,10);
hills[12].setDates(dates);
hills[12].setLatestDate(new GregorianCalendar(2016,0,10));
dates = new GregorianCalendar[4];
dates[0] = new GregorianCalendar(2005,1,11);
dates[1] = new GregorianCalendar(2005,1,12);
dates[2] = new GregorianCalendar(2008,11,13);
dates[3] = new GregorianCalendar(2008,11,14);
hills[13].setLatestDate(new GregorianCalendar(2008,11,14));
hills[13].setDates(dates);
dates = new GregorianCalendar[30];
dates[0] = new GregorianCalendar(2001,0,27);
dates[1] = new GregorianCalendar(2001,0,28);
dates[2] = new GregorianCalendar(2002,0,26);
dates[3] = new GregorianCalendar(2002,0,27);
dates[4] = new GregorianCalendar(2003,0,25);
dates[5] = new GregorianCalendar(2003,0,26);
dates[6] = new GregorianCalendar(2004,0,24);
dates[7] = new GregorianCalendar(2004,0,25);
dates[8] = new GregorianCalendar(2005,1,5);
dates[9] = new GregorianCalendar(2005,1,6);
dates[10] = new GregorianCalendar(2006,0,21);
dates[11] = new GregorianCalendar(2006,0,22);
dates[12] = new GregorianCalendar(2008,1,2);
dates[13] = new GregorianCalendar(2008,1,3);
dates[14] = new GregorianCalendar(2009,0,31);
dates[15] = new GregorianCalendar(2009,1,1);
dates[16] = new GregorianCalendar(2010,0,16);
dates[17] = new GregorianCalendar(2010,0,17);
dates[18] = new GregorianCalendar(2011,0,15);
dates[19] = new GregorianCalendar(2011,0,16);
dates[20] = new GregorianCalendar(2012,0,28);
dates[21] = new GregorianCalendar(2012,0,29);
dates[22] = new GregorianCalendar(2013,0,19);
dates[23] = new GregorianCalendar(2013,0,20);
dates[24] = new GregorianCalendar(2014,0,25);
dates[25] = new GregorianCalendar(2014,0,26);
dates[26] = new GregorianCalendar(2015,0,24);
dates[27] = new GregorianCalendar(2015,0,25);
dates[28] = new GregorianCalendar(2016,0,30);
dates[29] = new GregorianCalendar(2016,0,31);
hills[14].setLatestDate(new GregorianCalendar(2016,0,31));
hills[14].setDates(dates);
dates = new GregorianCalendar[11];
dates[0] = new GregorianCalendar(2007,0,13);
dates[1] = new GregorianCalendar(2009,2,15);
dates[2] = new GregorianCalendar(2011,1,12);
dates[3] = new GregorianCalendar(2011,1,13);
dates[4] = new GregorianCalendar(2013,0,26);
dates[5] = new GregorianCalendar(2013,0,27);
dates[6] = new GregorianCalendar(2015,1,14);
dates[7] = new GregorianCalendar(2015,1,15);
dates[8] = new GregorianCalendar(2016,1,12);
dates[9] = new GregorianCalendar(2016,1,13);
dates[10] = new GregorianCalendar(2016,1,14);
hills[15].setDates(dates);
hills[15].setLatestDate(new GregorianCalendar(2016,1,14));
dates = new GregorianCalendar[27];
dates[0] = new GregorianCalendar(2000,11,2);
dates[1] = new GregorianCalendar(2000,11,2);
dates[2] = new GregorianCalendar(2004,2,12);
dates[3] = new GregorianCalendar(2005,2,11);
dates[4] = new GregorianCalendar(2005,11,3);
dates[5] = new GregorianCalendar(2005,11,4);
dates[6] = new GregorianCalendar(2006,2,10);
dates[7] = new GregorianCalendar(2006,11,2);
dates[8] = new GregorianCalendar(2006,11,3);
dates[9] = new GregorianCalendar(2007,2,16);
dates[10] = new GregorianCalendar(2008,2,7);
dates[11] = new GregorianCalendar(2009,11,5);
```

```
dates[12] = new GregorianCalendar(2009,11,6);
dates[13] = new GregorianCalendar(2009,2,13);
dates[14] = new GregorianCalendar(2010,2,12);
dates[15] = new GregorianCalendar(2010,11,4);
dates[16] = new GregorianCalendar(2010,11,5);
dates[17] = new GregorianCalendar(2011,11,3);
dates[18] = new GregorianCalendar(2011,11,4);
dates[19] = new GregorianCalendar(2012,10,24);
dates[20] = new GregorianCalendar(2012,10,25);
dates[21] = new GregorianCalendar(2013,11,7);
dates[22] = new GregorianCalendar(2013,11,8);
dates[23] = new GregorianCalendar(2014,11,6);
dates[24] = new GregorianCalendar(2014,11,7);
dates[25] = new GregorianCalendar(2015,11,5);
dates[26] = new GregorianCalendar(2015,11,6);
hills[16].setLatestDate(new GregorianCalendar(2015,11,6));
hills[16].setDates(dates);
dates = new GregorianCalendar[17];
dates[0] = new GregorianCalendar(2001,2,11);
dates[1] = new GregorianCalendar(2002,2,17);
dates[2] = new GregorianCalendar(2003,2,8);
dates[3] = new GregorianCalendar(2003,2,9);
dates[4] = new GregorianCalendar(2004,2,14);
dates[5] = new GregorianCalendar(2005,2,13);
dates[6] = new GregorianCalendar(2006,2,12);
dates[7] = new GregorianCalendar(2007,2,17);
dates[8] = new GregorianCalendar(2007,2,18);
dates[9] = new GregorianCalendar(2008,2,9);
dates[10] = new GregorianCalendar(2010,2,14);
dates[11] = new GregorianCalendar(2012,2,11);
dates[12] = new GregorianCalendar(2013,2,17);
dates[13] = new GregorianCalendar(2014,2,9);
dates[14] = new GregorianCalendar(2015,2,14);
dates[15] = new GregorianCalendar(2015,2,15);
dates[16] = new GregorianCalendar(2016,1,6);
hills[17].setLatestDate(new GregorianCalendar(2016,1,6));
hills[17].setDates(dates);
dates = new GregorianCalendar[4];
dates[0] = new GregorianCalendar(2013,0,9);
dates[1] = new GregorianCalendar(2014,0,16);
dates[2] = new GregorianCalendar(2015,0,15);
dates[3] = new GregorianCalendar(2016,2,4);
hills[18].setDates(dates);
hills[18].setLatestDate(new GregorianCalendar(2016,2,4));
dates = new GregorianCalendar[31];
dates[0] = new GregorianCalendar(2002,0,19);
dates[1] = new GregorianCalendar(2002,0,20);
dates[2] = new GregorianCalendar(2003,0,18);
dates[3] = new GregorianCalendar(2003,0,19);
dates[4] = new GregorianCalendar(2004,0,11);
dates[5] = new GregorianCalendar(2004,0,18);
dates[6] = new GregorianCalendar(2005,0,29);
dates[7] = new GregorianCalendar(2005,0,30);
dates[8] = new GregorianCalendar(2006,0,28);
dates[9] = new GregorianCalendar(2006,0,29);
dates[10] = new GregorianCalendar(2007,0,20);
dates[11] = new GregorianCalendar(2007,0,21);
dates[12] = new GregorianCalendar(2008,0,25);
dates[13] = new GregorianCalendar(2008,0,27);
dates[14] = new GregorianCalendar(2009,0,16);
dates[15] = new GregorianCalendar(2009,0,17);
dates[16] = new GregorianCalendar(2010,0,22);
dates[17] = new GregorianCalendar(2010,0,23);
dates[18] = new GregorianCalendar(2011,0,21);
dates[19] = new GregorianCalendar(2011,0,22);
dates[20] = new GregorianCalendar(2011,0,23);
dates[21] = new GregorianCalendar(2012,0,20);
dates[22] = new GregorianCalendar(2012,0,21);
dates[23] = new GregorianCalendar(2013,0,11);
dates[24] = new GregorianCalendar(2013,0,12);
dates[25] = new GregorianCalendar(2014,0,18);
dates[26] = new GregorianCalendar(2014,0,19);
dates[27] = new GregorianCalendar(2015,0,17);
dates[28] = new GregorianCalendar(2015,0,18);
```

```
dates[29] = new GregorianCalendar(2016,0,23);
dates[30] = new GregorianCalendar(2016,0,24);
hills[19].setLatestDate(new GregorianCalendar(2016,0,24));
hills[19].setDates(dates);
dates = new GregorianCalendar[32];
dates[0] = new GregorianCalendar(2001,2,17);
dates[1] = new GregorianCalendar(2001,2,18);
dates[2] = new GregorianCalendar(2002,2,23);
dates[3] = new GregorianCalendar(2003,2,21);
dates[4] = new GregorianCalendar(2003,2,22);
dates[5] = new GregorianCalendar(2003,2,23);
dates[6] = new GregorianCalendar(2005,2,19);
dates[7] = new GregorianCalendar(2005,2,20);
dates[8] = new GregorianCalendar(2006,2,18);
dates[9] = new GregorianCalendar(2006,2,19);
dates[10] = new GregorianCalendar(2007,2,23);
dates[11] = new GregorianCalendar(2007,2,24);
dates[12] = new GregorianCalendar(2007,2,25);
dates[13] = new GregorianCalendar(2008,2,14);
dates[14] = new GregorianCalendar(2008,2,15);
dates[15] = new GregorianCalendar(2008,2,16);
dates[16] = new GregorianCalendar(2009,2,20);
dates[17] = new GregorianCalendar(2009,2,21);
dates[18] = new GregorianCalendar(2009,2,22);
dates[19] = new GregorianCalendar(2011,2,18);
dates[20] = new GregorianCalendar(2011,2,19);
dates[21] = new GregorianCalendar(2011,2,20);
dates[22] = new GregorianCalendar(2012,2,16);
dates[23] = new GregorianCalendar(2012,2,17);
dates[24] = new GregorianCalendar(2012,2,18);
dates[25] = new GregorianCalendar(2013,2,23);
dates[26] = new GregorianCalendar(2014,2,21);
dates[27] = new GregorianCalendar(2014,2,22);
dates[28] = new GregorianCalendar(2014,2,23);
dates[29] = new GregorianCalendar(2015,2,20);
dates[30] = new GregorianCalendar(2015,2,21);
dates[31] = new GregorianCalendar(2015,2,22);
hills[20].setDates(dates);
hills[20].setLatestDate(new GregorianCalendar(2015,2,22));
dates = new GregorianCalendar[3];
dates[0] = new GregorianCalendar(2011,2,7);
dates[1] = new GregorianCalendar(2002,2,12);
dates[2] = new GregorianCalendar(2014,1,26);
hills[21].setDates(dates);
hills[21].setLatestDate(new GregorianCalendar(2014,1,26));
dates = new GregorianCalendar[31];
dates[0] = new GregorianCalendar(2000,11,16);
dates[1] = new GregorianCalendar(2000,11,17);
dates[2] = new GregorianCalendar(2001,11,15);
dates[3] = new GregorianCalendar(2001,11,16);
dates[4] = new GregorianCalendar(2003,11,20);
dates[5] = new GregorianCalendar(2004,11,18);
dates[6] = new GregorianCalendar(2004,11,19);
dates[7] = new GregorianCalendar(2005,11,17);
dates[8] = new GregorianCalendar(2005,11,18);
dates[9] = new GregorianCalendar(2006,11,16);
dates[10] = new GregorianCalendar(2006,11,17);
dates[11] = new GregorianCalendar(2007,11,22);
dates[12] = new GregorianCalendar(2007,11,23);
dates[13] = new GregorianCalendar(2008,11,20);
dates[14] = new GregorianCalendar(2008,11,21);
dates[15] = new GregorianCalendar(2009,11,18);
dates[16] = new GregorianCalendar(2009,11,19);
dates[17] = new GregorianCalendar(2009,11,20);
dates[18] = new GregorianCalendar(2010,11,17);
dates[19] = new GregorianCalendar(2010,11,18);
dates[20] = new GregorianCalendar(2010,11,19);
dates[21] = new GregorianCalendar(2011,11,17);
dates[22] = new GregorianCalendar(2011,11,18);
dates[23] = new GregorianCalendar(2012,11,15);
dates[24] = new GregorianCalendar(2012,11,16);
dates[25] = new GregorianCalendar(2013,11,21);
dates[26] = new GregorianCalendar(2013,11,22);
dates[27] = new GregorianCalendar(2014,11,20);
```

```
                dates[28] = new GregorianCalendar(2014,11,21);
                dates[29] = new GregorianCalendar(2015,11,19);
                dates[30] = new GregorianCalendar(2015,11,20);
                hills[22].setDates(dates);
                hills[22].setLatestDate(new GregorianCalendar(2015,11,20));
                dates = new GregorianCalendar[3];
                dates[0] = new GregorianCalendar(2001,0,19);
                dates[1] = new GregorianCalendar(2001,0,20);
                dates[2] = new GregorianCalendar(2004,1,28);
                hills[23].setDates(dates);
                hills[23].setLatestDate(new GregorianCalendar(2004,1,28));
                for(int m = 0; m < hills.length; m++) {
                        System.out.println(m + " " +
                hills[m].getLatestDate().get(GregorianCalendar.YEAR));
                }
                PrintWriter out;
                Calendar c = new GregorianCalendar();
                c.set(Calendar.HOUR_OF_DAY, 0);
                try {
                        out = new PrintWriter("longest_jumps_rows.txt");
                        out.println("INSERT ALL");
                        double record = 0;
                        String tempDate = "";
                        SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MMM-yyyy");
                        int counter = 1;
                        hillsNum = new TreeSet<Integer>();
                        for (int i = 0; i < skiJumpers.length; i++) {
                                unattended = new ArrayList<Integer>();
                                for (int k = 0; k < 5; k++) {
                                        unattended.add(random.nextInt(24));
                                }
                                for (int j = 0; j < hills.length; j++) {
                                        id = j + 1;

        if(hills[j].getLatestDate().compareTo(skiJumpers[i].getStartDate()) == -1) {
                                                continue;
                                        }
                                        if (skiJumpers[i].getRank() < 7 || (skiJumpers[i].getRank() <
8 && c.get(GregorianCalendar.YEAR)
                                                        -
skiJumpers[i].getStartDate().get(GregorianCalendar.YEAR) < 4)) {
                                                if (unattended.contains(j)) {
                                                        continue;
                                                }
                                        }
                                        int pos = 0;
                                        Calendar tempCal = new GregorianCalendar();
                                        do {
                                                pos = random.nextInt(hills[j].getDates().length);
                                                tempCal = hills[j].getDates()[pos];
                                                System.out.println("hill: " + j + " pos " + pos + " "
+ tempCal.get(GregorianCalendar.YEAR));
                                                System.out.println("sj " + i);
                                                dateFormat.setTimeZone(tempCal.getTimeZone());
                                                tempDate = dateFormat.format(tempCal.getTime());
                                                System.out.println(tempDate);
                                        } while (tempCal.compareTo(skiJumpers[i].getStartDate()) ==
-1);
                                        hillsNum.add(j);
                                        if (skiJumpers[i].getRank() == 6) {
                                                if (hills[j].getkPoint() < 135) {
                                                        if(hills[j].getRecord() < 136) {
                                                                record = 100 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 114))) / 2
                                                                        + (double) random.nextInt(2)
/ 2;
                                                        } else {
                                                                record = 100 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 109))) / 2
                                                                        + (double)
random.nextInt(2) / 2;
                                                        }
                                                } else if (hills[j].getkPoint() == 185) {
```

```
                                                record = 160 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 155))) / 2
                                                        + (double) random.nextInt(2)
/ 2;
                                        } else if (hills[j].getkPoint() > 195) {
                                                record = 160 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 160))) / 2
                                                        + (double) random.nextInt(2)
/ 2;
                                        }
                                } else if (skiJumpers[i].getRank() == 7) {
                                        if (hills[j].getkPoint() < 135) {
                                                if(hills[j].getRecord() < 136) {
                                                        record = 110 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 122))) / 2
                                                                + (double) random.nextInt(2)
/ 2;
                                                } else {
                                                        record = 110 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 117))) / 2
                                                                        + (double)
random.nextInt(2) / 2;
                                                }
                                        } else if (hills[j].getkPoint() == 185) {
                                                record = 180 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 170))) / 2
                                                        + (double) random.nextInt(2)
/ 2;
                                        } else if (hills[j].getkPoint() > 195) {
                                                if (skiJumpers[i].getId() == 8 ||
skiJumpers[i].getId() == 13 || skiJumpers[i].getId() == 85
                                                                || skiJumpers[i].getId() ==
86 || skiJumpers[i].getId() == 19) {
                                                        record = 220 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 180))) / 2
                                                                        + (double)
random.nextInt(6) / 2;
                                                } else {
                                                        record = 180 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 170))) / 2
                                                                        + (double)
random.nextInt(2) / 2;
                                                }
                                        }
                                } else if (skiJumpers[i].getRank() == 8) {
                                        if (hills[j].getkPoint() < 135) {
                                                if(hills[j].getRecord() < 136) {
                                                        record = 110 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 127))) / 2
                                                                + (double) random.nextInt(2)
/ 2;
                                                } else {
                                                        record = 120 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 127))) / 2
                                                                        + (double)
random.nextInt(2) / 2;
                                                }
                                        } else if (hills[j].getkPoint() == 185) {
                                                record = 190 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 170))) / 2
                                                        + (double) random.nextInt(2)
/ 2;
                                        } else if (hills[j].getkPoint() > 195) {
                                                if (skiJumpers[i].getId() == 8 ||
skiJumpers[i].getId() == 13 || skiJumpers[i].getId() == 85
                                                                || skiJumpers[i].getId() ==
86 || skiJumpers[i].getId() == 19) {
                                                        record = 220 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 180))) / 2
                                                                        + (double)
random.nextInt(6) / 2;
                                                } else {
                                                        record = 190 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 170))) / 2
```

```
                                                                        + (double)
random.nextInt(2) / 2;
                                                        }
                                                }
                                        } else if (skiJumpers[i].getRank() == 9) {
                                                if (hills[j].getkPoint() < 135) {
                                                        if(hills[j].getRecord() < 136) {
                                                                record = 115 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 120))) / 2
                                                                        + (double) random.nextInt(4)
/ 2;
                                                        } else {
                                                                record = 125 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 130))) / 2
                                                                        + (double)
random.nextInt(4) / 2;
                                                        }
                                                } else if (hills[j].getkPoint() == 185) {
                                                        record = 200 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 180))) / 2
                                                                + (double) random.nextInt(4)
/ 2;
                                                } else if (hills[j].getkPoint() > 195) {
                                                        if (skiJumpers[i].getId() == 8 ||
skiJumpers[i].getId() == 13 || skiJumpers[i].getId() == 85
                                                                || skiJumpers[i].getId() ==
86 || skiJumpers[i].getId() == 19) {
                                                                record = 220 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 180))) / 2
                                                                        + (double)
random.nextInt(6) / 2;
                                                        } else {
                                                                record = 200 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 170))) / 2
                                                                        + (double)
random.nextInt(4) / 2;
                                                        }
                                                }
                                        } else if (skiJumpers[i].getRank() == 10) {
                                                if (hills[j].getkPoint() < 135) {
                                                        if(hills[j].getRecord() < 136) {
                                                                record = 120 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 125))) / 2
                                                                        + (double)
random.nextInt(6) / 2;
                                                        } else {
                                                                record = 130 + (double) (2 *
random.nextInt((int) (hills[j].getRecord() - 134))) / 2
                                                                        + (double)
random.nextInt(6) / 2;
                                                        }
                                                } else if (hills[j].getkPoint() == 185) {
                                                        record = 200 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 175))) / 2
                                                                + (double) random.nextInt(6)
/ 2;
                                                } else if (hills[j].getkPoint() > 195) {
                                                        record = 220 + (double) (2 *
random.nextInt((int) (hills[j].getkPoint() - 180))) / 2
                                                                + (double) random.nextInt(6)
/ 2;
                                                }
                                        }
                                        out.println("\tINTO T_LONGEST_JUMPS(jump_id, ski_jumper_id,
hill_id, date_taken, distance)");
                                        out.println("   VALUES(" + counter + ", " +
skiJumpers[i].getId() +", " + id +", '" + tempDate + "', " + record + ")");
                                        counter++;
                                }
                        }
                        out.println("SELECT 1 FROM DUAL;");
                        out.close();
                } catch (FileNotFoundException e) {
                }
```

```
            System.out.println(hillsNum);
        }
}
```

II. Implementation changes

a. Changes in tables

- Countries table:
    - instead of number_of_wins - no_of_winter_og_medals attribute is used (makes more sense)
    - instead of no_of_reg_ski_jumpers - no_of_active_ski_jumpers (those, who can start in WC) is implemented - it is not feasible to gather info about all registered ski jumpers
- Coaches table:
    - phone_number is now nullable
- Coaching_job_assignments table:
    - job_finished is now NULL by default

b. Changes in queries

- Query: "All ski jumpers, who don't have a hill in their country and yet managed to won over 10 World Cup competitions (first_name, last_name, name (Country), world_cup_wins)":
    - has been changed so that having no hill is not taken into consideration, when returning ski jumpers with over 10 WC wins
- Query: "Ski jumper with no official hill record and over 20 wins (first_name, last_name, name (Country), world_cup_wins)":
    - has been changed to no record and most wins
- Query: "Ski jumper with country of population below 30 million and less than 3 hills, who won the most WCs (first_name, last_name, name (Country), hill count, world_cup_wins)":
    - has been changed such that hill count is not displayed
- Query: "All ski jumpers, which have all their longest jumps of length over 105% of the critical point of a hill (first_name (Ski Jumper), last_name (Ski_Jumper), date_taken (Longest_Jump), distance (Longest_Jump), critical_point (Hill))":
    - has been replaced by query displaying min, avg and max age of ski jumper debuting in World Cup competition

III. Queries explained

a. Hill Records

Code:

```
SELECT (s.first_name||' '||s.last_name) as "RECORD HOLDER", distance, date_taken
as "DATE", city, hill_size

 FROM hills h join longest_jumps a on (h.hill_id = a.hill_id) join ski_jumpers s on
(a.ski_jumper_id = s.ski_jumper_id)

WHERE a.distance = (SELECT MAX(b.distance)

                    FROM longest_jumps b

                   WHERE a.hill_id = b.hill_id);
```

Explanation:

ski jumper first name and last name are displayed under the heading "RECORD HOLDER", date taken is displayed as "DATE", distance and hill_size are displayed as well

There are two parts of the query. in first, which could have ended before first WHERE keyword, all longest jumps of all ski jumpers are returned from three joined tables (hills and longest jumps on hill_id + longest_jumps and ski_jumpers on ski_jumper_id)

first WHERE clause passes distance and checks for match in the subquery, which returns max distance per hill id

b. Hill Records set by non-citizens of Country, in which the hill is

Code:

```
SELECT city, critical_point as "K POINT", hill_size as "HILL SIZE", (s.first_name||'
'||s.last_name) as "RECORD HOLDER", date_taken as "DATE", distance

 FROM hills h join longest_jumps a on (h.hill_id = a.hill_id) join ski_jumpers s on
(a.ski_jumper_id = s.ski_jumper_id)

WHERE a.distance = (SELECT MAX(b.distance)

                    FROM longest_jumps b
```

```
WHERE a.hill_id = b.hill_id) and h.country_code <>
s.country_code;
```

Explanation:

This query as similar to the one above, but additionally there is a check made, which will make sure that there is no match between country_code of a ski jumper and the hill

c. All ski jumpers from specified country

Code:

```
SELECT (REPLACE(first_name, SUBSTR(first_name, 2))||'. '|| last_name) as "SKI
JUMPER", date_of_birth AS "DATE OF BIRTH", world_cup_wins AS "WC WINS"
 FROM ski_jumpers NATURAL JOIN countries
WHERE name =:enter_country_name
ORDER BY last_name;
```

Explanation:

This query returns four columns per ski jumper. All ski jumpers from the country, which needs to be input, when the query is run. the output is sorted in ascending order by last name of ski jumper. In order to get the country name and map it with the ski_jumper - countries and ski jumpers tables need to be join (natural join is used in this case). First name of a ski jumper if trimmed to be displayed only as an initial along with a "." and last name as a "SKI JUMPER"

d. All ski jumpers with over 10 WC wins

Code:

```
SELECT (first_name||' '||last_name) as "COMPETITOR",
    name as "COUNTRY",
    world_cup_wins AS "WC WINS"
 FROM ski_jumpers NATURAL JOIN countries
WHERE world_cup_wins > 10
ORDER BY WORLD_CUP_WINS DESC;
```

Explanation:

Two tables: ski_jumpers and countries are joined using natural join.

First and last name of each ski jumper returned in the query is displayed under the heading "COMPETITOR", name of the country uses heading "COUNTRY", and world_cup_wins as "WC WINS". Number of world cup wins is restricted, such that

only ski jumpers with no of wins greater than 10 are displayed, sorted in descending order by world_cup_wins

e. Average, min and max age of WC debut among all ski jumpers

Code:

```
SELECT
(ROUND(AVG((first_start_date - date_of_birth) / 365))||' YEARS OF AGE')
AS "AVG WC DEBUT AGE",
(ROUND(MIN(first_start_date - date_of_birth) / 365)||' YEARS OF AGE')
AS "MIN WC DEBUT AGE",
(ROUND(MAX(first_start_date - date_of_birth) / 365)||' YEARS OF AGE')
AS "MAX_WC_DEBUT_AGE"
 FROM SKI_JUMPERS;
```

Explanation:

This query displays rounded age of the youngest and oldest ski jumper's debut in world cup competition and the average age of first start in world cup among all ski jumpers

f. 5 ski jumpers with most WC wins

Code:

```
SELECT *
 FROM (SELECT (first_name||' '||last_name) as "SKI JUMPER", name as
"COUNTRY", world_cup_wins as "WC WINS" FROM ski_jumpers NATURAL
JOIN countries ORDER BY world_cup_wins DESC)
 WHERE ROWNUM <= 5;
```

Explanation:

THe query inside the brackets prints all ski jumpers, with their names, country they represent and world cup wins.

Having this query in those brackets and selecting all from it with addition of "ROWNUM" limits the number of displayed records to whatever parameter is specified. in this case only the 5 ski jumpers with the greatest number of WC wins will be displayed

g. All jumps over 200m

Code:

```
SELECT (first_name||' '||last_name) as "SKI JUMPER", name as "COUNTRY", city,
distance, hill_size AS "HILL SIZE", date_taken as "DATE"

 FROM hills h join longest_jumps l on (h.hill_id = l.hill_id) join ski_jumpers s on
(l.ski_jumper_id = s.ski_jumper_id) join countries c on (s.country_code =
c.country_code)

 WHERE distance > 200

 ORDER BY distance desc;
```

Explanation:

This query joins four tables on their primary/foreign keys using natural joins and displays first and last name as "SKI JUMPER", country name as "COUNTRY", city (location of the hill), hill size and date of the attempt and restricts the output to display only jumps over 200m. Printed records are sorted in descending order by the distance of a given jump

h. All ski jumpers cooperating with one coach for 5 or more years

Code:

```
SELECT (s.first_name||' '||s.last_name) as "SKI JUMPER", (c.first_name||'
'||c.last_name) as "COACH", (round((t.finish_date - t.start_date)/365)||' Years') as
"DURATION"

 FROM ski_jumpers s join training_history t ON (s.ski_jumper_id = t.ski_jumper_id)
join coaches c ON (c.coach_id = t.coach_id)

 WHERE t.finish_date - t.start_date > 1800

 ORDER BY t.finish_date - t.start_date desc;
```

Explanation:

This query joins 3 tables and returns full names of all ski jumpers and coaches and the time they have been training together as long as this time is not smaller than 5 years (in this query the number of days is rounded to 1800). results are ordered from these with the highest duration to the lowest

i. Ski jumper with no official hill record and most wins

Code:

```
SELECT * FROM

(SELECT (first_name||' '||last_name) as "SKI JUMPER", name as "COUNTRY",
world_cup_wins as "WC WINS"

  FROM SKI_JUMPERS NATURAL JOIN COUNTRIES

 WHERE SKI_JUMPER_ID NOT IN

(SELECT SKI_JUMPER_ID

  FROM LONGEST_JUMPS

 WHERE (DISTANCE, HILL_ID) IN

(SELECT MAX(distance), hill_id

  FROM longest_jumps

  GROUP BY hill_id))

  ORDER BY world_cup_wins desc)

WHERE ROWNUM <= 1;
```

Explanation:

The innermost subquery (starting with SELECT SKI_JUMPER_ID) selects all record holders' ids and the result of that query is passed to the query, which joins SKI_JUMPERS and COUNTRIES tables and checked against all the ski jumpers ids. Only those ski jumpers, whose id don't match the innermost subquery, are returned. the output is narrowed down to 1 record by using ROWNUM <= 1

j. Ski jumper with small population country, less than 2 hills and most WC wins

Code:

```
SELECT * FROM

(SELECT (first_name||' '||last_name) as "SKI JUMPER", name as "COUNTRY",
world_cup_wins as "WC WINS"

FROM ski_jumpers NATURAL JOIN countries NATURAL JOIN hills

NATURAL JOIN
```

```
(SELECT * FROM (SELECT COUNT(HILL_ID) AS "HILL_COUNT",
COUNTRY_CODE

FROM HILLS

GROUP BY COUNTRY_CODE

ORDER BY HILL_COUNT DESC)

WHERE HILL_COUNT < 3)

WHERE POPULATION < 30000000

ORDER BY WORLD_CUP_WINS DESC)

WHERE ROWNUM <=1;
```

Explanation:

In this query, as a first step, hill counts per country are returned in descending order and then those results are narrowed down to those countries, which have less than 3 hills. In next step only ski jumpers, which represent countries which have less than 3 hills and additionally population less than 30M, are returned (in descending order by world cup wins). Finally, only the ski jumper, which has the most world cup wins is returned

k. Ski jumper with no WC win from the top OG medals countries

Code:

```
SELECT (first_name||' '||last_name) as "SKI JUMPER", NAME AS "COUNTRY", ('--
----------_' ||NO_OF_WINTER_OG_MEDALS||'_----------') AS "OLYMPIC
GAMES MEDALS"

 FROM SKI_JUMPERS NATURAL JOIN COUNTRIES

WHERE NO_OF_WINTER_OG_MEDALS > 200 AND WORLD_CUP_WINS = 0

ORDER BY COUNTRY_CODE;
```

Explanation:

In this query all ski jumpers from the countries with number of winter OG medals above 200 are displayed in the following manner:

full ski jumper's name is displayed as "SKI JUMPER", his country of origin as "COUNTRY", number of medals awarded (as "OLYMPIC GAMES MEDALS" by the country (padding with hyphens and underscores is used to centre those numbers)

l. All hills with hill size over 200m

Code:

```
SELECT name as "COUNTRY", city, hill_size as "HILL SIZE (IN METRES)",
max(distance) AS "RECORD"

 FROM HILLS T JOIN LONGEST_JUMPS L ON (T.HILL_ID = L.HILL_ID) JOIN
countries c ON (t.country_code = c.country_code)

 WHERE t.hill_size > 200

 GROUP BY hill_size, name, city;
```

Explanation:

This query returns all rows from three tables joined on their primary / foreing keys relationships including all hills, whose hill size is greater than 200m. Three aliases are used as headings for name of the country (COUNTRY), hill_size (HILL SIZE IN METERS)), max(distance) (RECORD).

m. Coach hired by most federations

Code:

```
SELECT (first_name||' '||last_name) AS "COACH", name as "WORKING FOR",
job_started as "WORKING FROM", job_finished as "WORKING TO"

 FROM coaching_job_assignments natural join coaches natural join countries

where coach_id = (

SELECT cnt1.coach_id

 FROM (SELECT COUNT(*) as total, coach_id

       FROM coaching_job_assignments

      GROUP BY coach_id) cnt1,

   (SELECT MAX(total) as maxtotal
```

```
    FROM (SELECT COUNT(*) as total, coach_id from coaching_job_assignments
group by coach_id)) cnt2

 WHERE cnt1.total = cnt2.maxtotal and rownum <=1)

 ORDER BY "WORKING FROM";
```

Explanation:

This query in order to work needed to be broken down to several subqueries. Firstly all coach-ids are returned along with the count of their job assignments, then the result is limited to the coach with max number of job assignments and his coach_id is returned. Finally details gathered from 3 tables joined using natural join (all coaches are matched with previously returned id (one id to be precise) and the matched coach's details (his name, countries he worked for and start / finish date) are returned

n. Coach hired for the longest period of time by a federation

Code:

```
SELECT *

  FROM (SELECT (first_name||' '||last_name) AS "COACH", job_started,
job_finished, name as "COUNTRY"

        FROM coaches T JOIN coaching_job_assignments a ON (T.coach_ID =
a.coach_ID) JOIN countries c ON (a.country_code = c.country_code)

          ORDER BY job_started - job_finished)

 WHERE ROWNUM <= 1;
```

Explanation:

The subquery returns details of the longest job assignment from 3 joined tables. The final output is the subquery limited to the longest job assignment

o. Country with the biggest number of hills

Code:

```
SELECT (name) as "COUNTRY", count(country_code) as "NUMBER OF HILLS"

  FROM countries NATURAL JOIN hills

 WHERE country_code = (

          SELECT cnt1.country_code
```

```
FROM (SELECT COUNT(*) as total, country_code
        FROM hills
        GROUP BY country_code) cnt1,
    (SELECT MAX(total) as maxtotal
     FROM (select COUNT(*) as total, country_code FROM hills group by country_code)) cnt2
        WHERE cnt1.total = cnt2.maxtotal)
 GROUP BY name;
```

Explanation:

First step of this query involves returning all country codes with the number of hills in given country. Next the country code with top number of hills is returned and finally the hill count and name of the country are returned

p. The shortest jump being hill record

Code:

```
SELECT * FROM
(SELECT distance, date_taken as "DATE", city, hill_size, (s.first_name||' '||s.last_name) as "RECORD_HOLDER"
  FROM hills h join longest_jumps a on (h.hill_id = a.hill_id) join ski_jumpers s on (a.ski_jumper_id = s.ski_jumper_id)
  WHERE a.distance = (SELECT MAX(b.distance)
                      FROM longest_jumps b
                      WHERE a.hill_id = b.hill_id)
  ORDER BY distance)
 WHERE ROWNUM <= 1;
```

Explanation:

This query returns first row from the subquery, which returns all hill records in ascending order by distance