Programming Systems and Environments, Lab 2

Jakub Grzana, 241530

Task 1 – Random Number Generator

I've expanded program from Lab1 by adding to Person class static attribute of type "SecureRandom", which as far as I know is random number generator safe for cryptographic purposes. I'm not using it with anything regarding security so it's massive overkill. Furthermore, I've added member functions to generate random age (which generates random number between specified min and max), random name and surname (those two simply get random element from predetermined list). I've added default constructor that sets age, name and surname to those random values.

Task 2 – People

I've made People class that contains ArrayList of People. Constructor with single integer parameter allows to create N people that are stored within that ArrayList, their data is generated randomly with methods described in Task 1. Overloaded toString method is used to print content of ArrayList. For sorting, java.util.Collections.sort is used, with comparator given in the form of lambda expression. I've made member functions to sort by age, salary, name, surname and fullname. To not bother myself with changing order of sorting (ascending/descending) I've added member function "reverse".

Task 3 – Extension (salary)

I've added private attribute "salary" to Person class, corresponding to it getter and setter methods, random value generated in default constructor and so on. In People class, I've added averageSalary method that calculates average salary of all people in the list, and selectSalary method that takes single argument (minimum salary) and selects only those people from the list that have salary above this value. So, to get solution to the problem mentioned in task description, first call selectSalary, then on the output use averageSalary. I believe this approach is more universal and useful than specialized member function.

Conclusions

Good to see lambda expressions that can consist of whole blocks of code rather than single instructions. Also, I've always wondered why string comparison returns integer instead of boolean – now I see this approach can be used to sort strings alphabetically.

CODE: Main.java

```
package Program;
public class Main {
    public static void main(String[] args) {
        // Person class task
        People p = new People(12);
        p.sortBySalary();
        p.reverse();

        System.out.println(p);
        System.out.println(p.selectSalary(5000).averageSalary());
    }
}
```

CODE: Person.java

```
package Program;
import java.lang.*;
import java.util.ArrayList;
import java.security.SecureRandom;

public class Person {
    private int age;
    private int salary;
    private String name;
    private String surname;
    private static SecureRandom randomGenerator = new SecureRandom();

    private void setAge(int age) { this.age = age; }
    public int getAge() { return this.age; }
    private void setName(String name) { this.name = name; }
    public String getName() { return this.name; }
    private void setSurname(String surname) { this.surname = surname; }
    public String getSurname() { return this.surname; }
    private void setSalary(int salary) { this.salary = salary; }
    public int getSalary() { return this.salary; }
```

```
private static String GetRandomName()
{
    ArrayList<String> names = new ArrayList<>();
    names.add("Jagoda");
    names.add("Eliza");
    names.add("Harold");
    names.add("Levi");
    names.add("Samuel");
    names.add("Igor");
    names.add("Ewelina");
    names.add("MUNDO");
    final int index = randomGenerator.nextInt(names.size());
    return names.get(index);
}

private static String GetRandomSurname()
{
    ArrayList<String> names = new ArrayList<>();
    names.add("Spirit");
    names.add("Orzeszkowa");
    names.add("Ackermann");
    names.add("Ortega");
    names.add("Light");
    names.add("Zawicki");
    names.add("Nerina");
    final int index = randomGenerator.nextInt(names.size());
    return names.get(index);
}

private static int GetRandomNum(int min, int max)
{
    final int diff = Math.abs(max - min);
    final int rand_val = randomGenerator.nextInt(diff);
    return min + rand_val;
}
```

```
    Person()
    {
        this.setName(GetRandomName());
        this.setSurname(GetRandomSurname());
        this.setAge(GetRandomNum(18,24));
        this.setSalary(GetRandomNum(10, 10000));
    }

    Person(int age, String name, String surname, int salary)
    {
        this.setAge(age);
        this.setName(name);
        this.setSurname(surname);
        this.setSalary(salary);
    }

    @Override public String toString()
    {
        return String.format("======\nName: %s\nSurname: %s\nAge: %d\nSalary: %d",
this.getName(), this.getSurname(), this.getAge(), this.getSalary());
    }
}
```

CODE: People.java

```
package Program;

import java.lang.Math;
import java.util.function.Consumer;
import java.util.*;

public class People {
   private List<Person> personList = new ArrayList<>();
   public List<Person> getPersonList() { return this.personList; }
   private void addPerson(Person p) { this.personList.add(p); }
   People() {}

   People(int n)
   {
      for(int i = 0; i < n; ++i)
      {
         this.addPerson(new Person());
      }
   }

   public void sort(Comparator<Person> cmp)
   {
      Collections.sort(this.getPersonList(), cmp);
   }

   public void sortByAge()
   {
      this.sort( (p1, p2) -> p1.getAge() - p2.getAge() );
   }

   public void sortBySalary()
   {
      this.sort( (p1, p2) -> p1.getSalary()- p2.getSalary() );
   }

   public void sortByName()
   {
      this.sort( (p1, p2) -> p1.getName().compareTo(p2.getName()) );
   }
```

```
public void sortBySurname()
{
   this.sort( (p1, p2) -> p1.getSurname().compareTo(p2.getSurname()) );
}

public void sortByNameAndSurname()
{
   this.sort( (p1, p2) -> {
      String n1 = String.format("%s %s", p1.getName(), p1.getSurname());
      String n2 = String.format("%s %s", p2.getName(), p2.getSurname());
      return n1.compareTo(n2);
   });
}

public void reverse()
{
     Collections.reverse(this.getPersonList());
}

public People selectFirst(int num)
{
   int mx = Math.min(num, this.getPersonList().size());
   People ppl = new People();
   for(int i = 0; i < mx; ++i)
   {
      ppl.addPerson(this.getPersonList().get(i));
   }
   return ppl;
}
```

```
    public People selectSalary(int salary)
    {
        People ppl = new People();
        this.sortBySalary();
        this.reverse();
        for(Person p : this.getPersonList())
        {
            if(p.getSalary() > salary)
            {
                ppl.addPerson(p);
            }
        }
        return ppl;
    }

    public double averageSalary()
    {
        double output = 0;
        for(Person p : this.getPersonList())
        {
            output = output + p.getSalary();
        }
        return output / this.getPersonList().size();
    }

    @Override public String toString()
    {
        String output = "";
        for(Person p : this.getPersonList())
        {
            output = output + p + "\n";
        }
        return output;
    }
}
```