Programmieren Aufgaben ab aula05

Aula05

5.1

Respecting the concepts of encapsulation, implement classes that allow you to model the following geometric shapes:

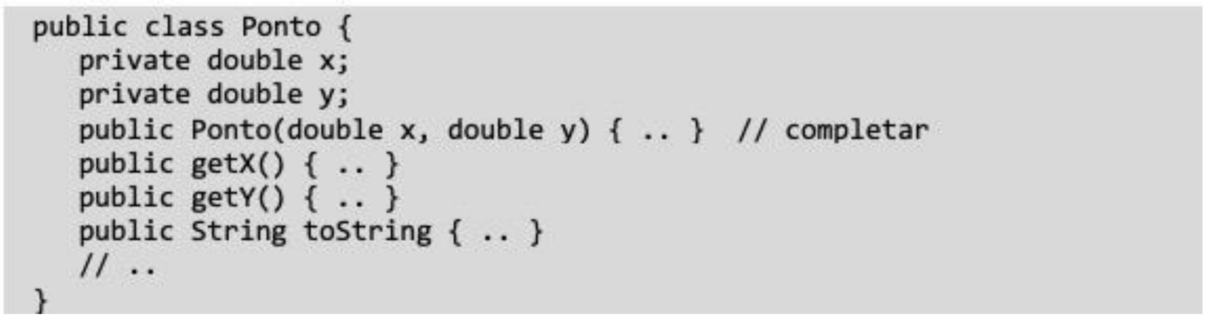
• Circle, characterized by a center and radius;

• Triangle, characterized by the dimension of its sides (side1, side2 and side3);

• Rectangle, characterized by length and height.

Also ensure the following specifications:

a) build the Ponto class;



```
public class Ponto {
   private double x;
   private double y;
   public Ponto(double x, double y) { .. }  // completar
   public getX() { .. }
   public getY() { .. }
   public String toString { .. }
   // ..
}
```

b) create classes that represent each of the geometric figures, implementing suitable constructors and methods for each class.

c) add all the important special methods (toString (), equals (), get ... (), set ... (), ...);

d) implement a method to calculate the area of each type of figure;

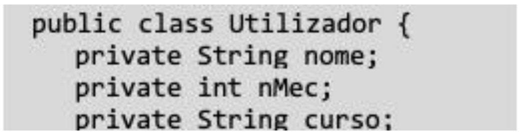
e) implement a method to calculate the perimeter of each type of figure;

f) implement a method to verify that the two circles intersect;

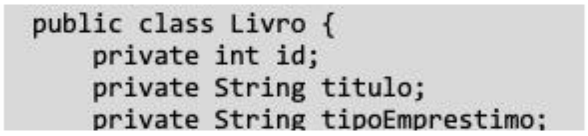
g) implement a program that allows you to test all created classes.

5.2

It is intended to build a simplified information system for the management of a university library. The library contains a book catalog and a set of users (students only). All users are identified by their mechanographic number, name and course. The books are characterized by an ID (numeric and sequential, starting at 100), title and type of loan (CONDITIONAL or NORMAL). Start with the following definitions:



```
public class Utilizador {
   private String nome;
   private int nMec;
   private String curso;
```



```
public class Livro {
    private int id;
    private String titulo;
    private String tipoEmprestimo;
```

Make use of access modifiers to ensure that all class attributes are not accessible from the outside. If necessary, define new attributes to meet the requirements of the statement. Test the classes developed with the following program:

```
import java.util.ArrayList;

public class Ex52 {

   public static void main(String[] args) {

      // Para o conjunto de Livros vamos criar um vetor de 10 posições
      // Este vetor tem uma dimensão fixa pelo que se for necessário guardar
      // mais livros, teremos de criar um vetor de maior dimensão.
      Livro catalogo[] = new Livro[10];
      catalogo[0] = new Livro("Java 8", "CONDICIONAL");
      catalogo[1] = new Livro("POO em Java 8");
      catalogo[2] = new Livro("Java para totós", "NORMAL");
      System.out.println("ID = " + catalogo[1].getId() + ", "
                              + catalogo[1].getTitulo());
      catalogo[2].setTipoEmprestimo("CONDICIONAL");

      for (int i = 0; i < catalogo.length; i++) { // usando o indice do vector
         if (catalogo[i] != null) // porque o vector catalogo não está cheio
            System.out.println(catalogo[i]);
      }

      // Para o conjunto de utilizadores usamos a classe java.util.ArrayList
      // É uma implementação de um vetor com tamanho variável
      ArrayList<Utilizador> alunos = new ArrayList<>();
      alunos.add(new Utilizador("Catarina Marques", 80232, "MIEGI"));
      alunos.add(new Utilizador("Joao Silva", 90123, "LEI"));
      alunos.get(1).setnMec(80123);

      for (Utilizador u : alunos) { // usando foreach
            System.out.println(u);
      }
   }
}
```

Whose result of its execution must be:

```
ID = 101, POO em Java 8
Livro 100; Java 8; CONDICIONAL
Livro 101; POO em Java 8; NORMAL
Livro 102; Java para totós; CONDICIONAL
Aluno: 80232; Catarina Marques; MIEGI
Aluno: 80123; Joao Silva; LEI
```

5.3
Using the classes developed in the previous exercise, implement a program that allows you to manage users and loans in a library. Start by building, in an interactive way, the following menu:
1 - enroll user
2 - remove user
3 - print user list
4 - register a new book
5 - print book list
6 - borrow
7 - return
8 - leave

Additional conditions:
a) It is recommended that loan and return operations be carried out based on the ID of the book and the student's mechanographic number.
b) Each student can only request a maximum of 3 books simultaneously. You must modify the user class to be able to save the IDs of the requested books, as well as the book class to indicate their availability.
c) For simplicity, consider that there is only one copy of each book and that books with a CONDITIONAL loan type cannot be requested.
d) To store the book catalog and the list of students, use vectors, considering that at most the library can have 100 books and 100 users.

aula06
6.1

Build the Person class which is characterized by name, citizen's card number and date of birth. Start with the following definitions and define new methods to include in the class's public interface.

```
public class Pessoa {
   private String nome;
   private int cc;
   private Data dataNasc;
   // .....
}
```

```
public class Data{
    private int dia;
    private int mes;
    private int ano;
    // .....
}
```

Create suitable methods to allow the initialization of your attributes when creating each object:

```
Data d = new Data(5, 10, 1988);
Pessoa p = new Pessoa("Ana Santos", 98012244, d);
```

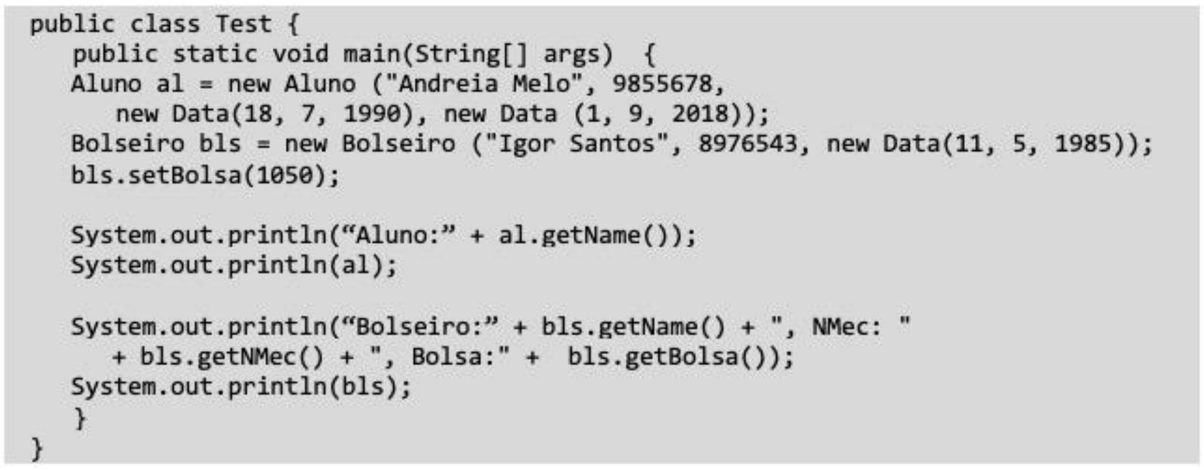
Build a new Student class, derived from the Person class, adding the necessary methods and attributes to access and store the mechanographic number (int) and the enrollment date (Date) at the educational institution. Note that the mechanographic number should be assigned automatically (and sequentially from 100) when creating a new student.
The simplified class structure should be as follows:

```
public class Pessoa {
   //...
   String getName(){...}         // retorna o nome da pessoa
}

public class Aluno extends Pessoa {
   //... definição de atributos

   Aluno(String iNome, int iBI, Data iDataNasc, Data iDataInsc);
   Aluno(String iNome, int iBI, Data iDataNasc);
            // nota: neste caso deve assumir a data atual
   int getNMec() {...}         // retorna o número mecanográfico
   // ... acrescentar métodos necessários
}
```

Create the Scholarship class, derived from the Student class, which should include an attribute with the monthly scholarship amount. Define new methods or rewrite the methods you see fit. Add get / set methods associated with the scholarship amount.

Implement the "@Override public String toString ()" method in all classes. For example, for the Person class, it should return:
"Ana Santos, CC: 98012244 Date: 5/10/1988"

Test the work done with the following program:



```
public class Test {
   public static void main(String[] args)  {
   Aluno al = new Aluno ("Andreia Melo", 9855678,
      new Data(18, 7, 1990), new Data (1, 9, 2018));
   Bolseiro bls = new Bolseiro ("Igor Santos", 8976543, new Data(11, 5, 1985));
   bls.setBolsa(1050);

   System.out.println("Aluno:" + al.getName());
   System.out.println(al);

   System.out.println("Bolseiro:" + bls.getName() + ", NMec: "
      + bls.getNMec() + ", Bolsa:" +  bls.getBolsa());
   System.out.println(bls);
   }
}
```

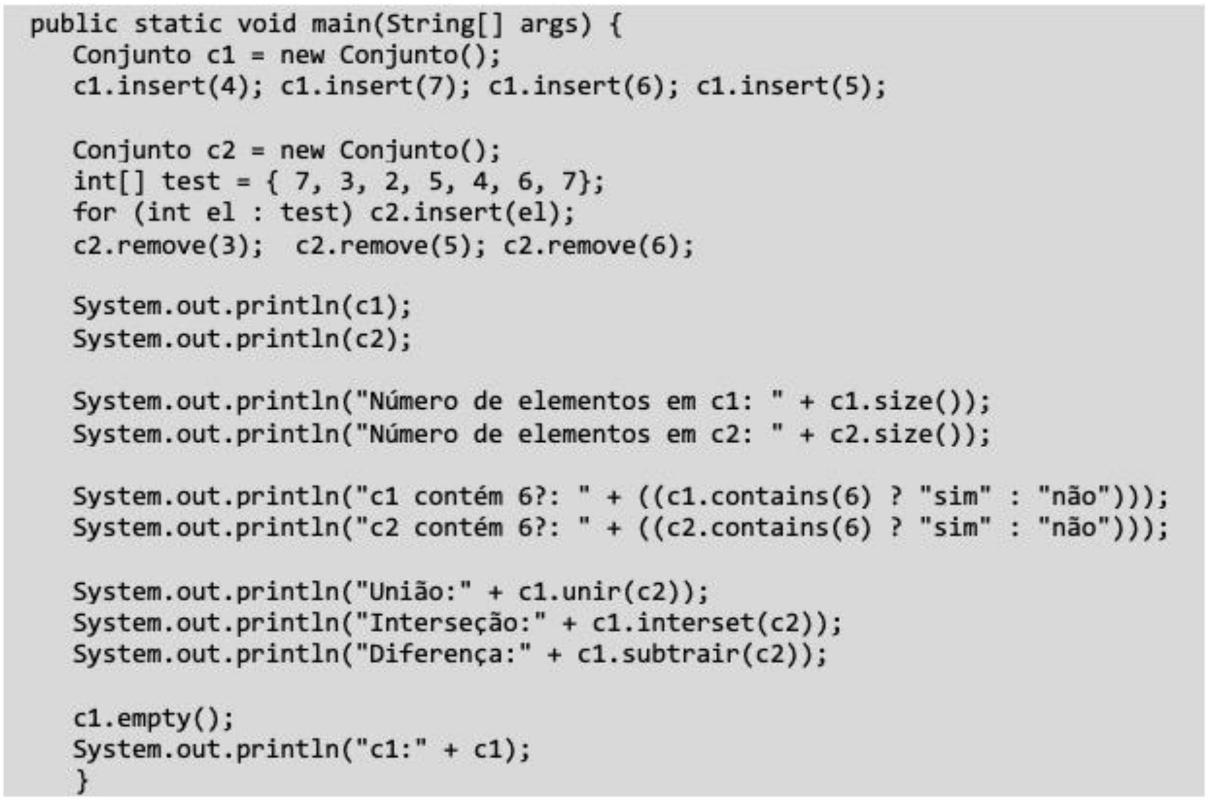
6.2
Using inheritance, rewrite the program developed in class 5 regarding geometric figures. Include a new attribute, color (String), common to all figures.

6.3

Build a Set class that holds a set of integers (which cannot be repeated). Use vectors and implement the following functions:

- void insert (int n); - to insert a new element in the set. If this element already exists, the function does nothing. Initially it is not known how many elements we will insert.
- boolean contains (int n); - to indicate whether a given element is in the set.
- voidremove (intn); - to remove an element from the set. If this element is not in the set, the function does nothing.
- void empty();-para apagar todos os elementos do conjunto.
- String toString (); - to convert the elements of the set into a String.
- int size (); - to calculate the number of elements in the set.
- Set set1 (Set set2); - to build a new set that represents the union of two sets. The resulting set must not contain elements repeated.
- Set subtract (Set diff) - to build a new set that represents the difference of this and the elements of the set represented by diff object.
- Set interset (Set inter) - to build a new set that represents the intersection of this with the elements of the set represented by the inter object. The resulting set cannot contain repeated elements.

Test the developed class with the following main function:



```java
public static void main(String[] args) {
   Conjunto c1 = new Conjunto();
   c1.insert(4); c1.insert(7); c1.insert(6); c1.insert(5);

   Conjunto c2 = new Conjunto();
   int[] test = { 7, 3, 2, 5, 4, 6, 7};
   for (int el : test) c2.insert(el);
   c2.remove(3);  c2.remove(5); c2.remove(6);

   System.out.println(c1);
   System.out.println(c2);

   System.out.println("Número de elementos em c1: " + c1.size());
   System.out.println("Número de elementos em c2: " + c2.size());

   System.out.println("c1 contém 6?: " + ((c1.contains(6) ? "sim" : "não")));
   System.out.println("c2 contém 6?: " + ((c2.contains(6) ? "sim" : "não")));

   System.out.println("União:" + c1.unir(c2));
   System.out.println("Interseção:" + c1.interset(c2));
   System.out.println("Diferença:" + c1.subtrair(c2));

   c1.empty();
   System.out.println("c1:" + c1);
   }
```

The results should be as follows (the order of the elements is irrelevant):

4 7 6 5

7 2 4

Number of elements in c1: 4 Number of elements in c2: 3 Does c1 contain 6 ?: yes

c2 contains 6 ?: no

Union: 4 7 6 5 2 Intersection: 7 4

Difference: 6 5

c1:

Aula07
7.1

It is intended to develop a program that allows the management of some products in a travel agency. The main entities in this information system are accommodation (apartments and hotel rooms) and cars that can be rented. The following characteristics must be supported:

- The travel agency, in addition to containing a set of accommodations and a set of rental vehicles, has a name (String) and an address (String);
- An accommodation has a code (String), name (String), location (String), price per night (double), availability (boolean) and rating (double, between 1.0 and 5.0). It must allow check-in and check-out operations.
- An apartment is an accommodation but has more information on the number of rooms.
- A hotel room is an accommodation but there is one more field that indicates the type (single, double, twin, triple).
- A car has class (char, from 'A' to 'F') and indications whether it is gasoline / diesel. It must allow pick-up and drop-off operations.

Properly represent all of these entities. Create constructors, set / get methods that seem appropriate, as well as methods that are essential for each class.
Test the developed classes using a program that simulates the interface with the agency's employee (for example, a menu), necessarily involving the creation of the various objects mentioned above. Simulate some booking and delivery operations and print out the current information about the agency at the end.

7.2

Consider the following entities that are part of a robotic football game:

- A moving object is characterized by coordinates (x and y) and the distance covered during the game.
  A moving object must allow the move operation (int newX, int newY). Whenever this operation is called, the distance traveled must be calculated automatically.
- Robot, is a mobile object characterized by an id (String), the type of player (a String that can be GuardaRedes, Avancado, Defense, Medio) and number of goals scored. A robot must also allow the operation to score a goal.
- Ball, movable object still characterized by a color (String);
- Team, characterized by a name (String), name of the person in charge (String), total of goals scored, total goals conceded and a set of robots.
- Game, characterized by information relating to two teams, a ball, game duration and elapsed time. Explore how you can automatically calculate elapsed time.

Properly represent all of these entities. Create constructors, set / get methods that seem appropriate, as well as methods that are essential for each class.
Test each of the developed classes by building a program for this purpose, involving the creation of the various objects. Evolve this program to allow you to simulate a game between two teams, each with 3 robots (e.g. simulate some movements, scoring goals, etc.).

Aula08

8.1

Consider the following entities and representative characteristics of motor vehicles:

- Motorcycle, characterized by license plate, make, model, cylinder capacity, type (sporty or road).
- Light car, characterized by registration, make, model, engine capacity, chassis number, luggage compartment capacity.
- Taxi, which has the characteristics of a light car and, in addition, the license number.
- Truck, characterized by registration number, make, model, cylinder capacity, chassis number, weight, maximum load.
- Bus, characterized by registration, make, model, cylinder capacity, chassis number, weight, maximum number of passengers.
- Car rental company, featuring a name, postal code, email and a set of vehicles, which can include any of the previous elements.

All vehicles must implement the following interface:

```
public interface KmPercorridosInterface {
   void trajeto(int quilometros);
   int ultimoTrajeto();
   int distanciaTotal();
}
```

a) Analyze the problem carefully and model the necessary interfaces and classes, their associations (inheritance, composition) as well as all attributes and methods. Build the main class, the main function, and write code to test various possibilities with the previous entities. For example:

- Create a company with one or more vehicles of each type.
- Test various methods (setters, getters, toString, equals, KmPinteridosInterface methods, ..).
- Indicate the vehicle with the most kilometers traveled.

b) (Open exercise) Add other features that seem interesting to you (for example, validating your registration, checking email format, sorting vehicles by cylinder capacity, etc.).

8.2

Consider the following entities and representative characteristics of food:

- Meat, has variety (cow, pork, turkey, chicken, other), proteins (double), calories (double), weight (double).
- Fish, has type (frozen or fresh), protein (double), calories (double), weight (double).
- Cereal, has a name (String), proteins (double), calories (double), weight (double). It is a vegetarian food.
- Vegetable, has name (String), protein (double), calories (double), weight (double). It is a vegetarian food.
- Dish, has a name (String) and composition (set of foods)
- Vegetarian Dish, has a name (String) and composition (set of foods vegetarians).
- PratoDieta, has a name (String) and composition (set of foods) and limit maximum calories (double).
- Menu - has a name (String), a location (String) and a list of dishes.

Analyze the problem carefully and model the necessary interfaces and classes, their associations (inheritance, composition) as well as all attributes and methods. Implement all necessary classes, following the following considerations:

- The unit of calories and protein is relative to 100gr.
- For each dish it must be possible to obtain information about food, total weight, calories, protein, ...
- Implement the hashCode (), equals (), toString () methods in all classes.
- The dishes must respect the java.lang.Comparable interface to allow for calorie-based comparison.

b) Test the implementation with program A08E02.java available in elearning.

aula09
9.1

The java.util.ArrayList <E> class is an implementation of a dynamic vector provided by JavaSE. Consult the documentation for this class and run the following program.

```java
import java.util.ArrayList;
import java.util.Collections;

public class ALDemo {

   public static void main(String[] args) {
      ArrayList<Integer> c1 = new ArrayList<>();
      for (int i = 10; i <= 100; i+=10)
         c1.add(i);
      System.out.println("Size: " + c1.size());
      for (int i = 0; i < c1.size(); i++)
         System.out.println("Elemento: " + c1.get(i));

      ArrayList<String> c2 = new ArrayList<>();
      c2.add("Vento");
      c2.add("Calor");
      c2.add("Frio");
      c2.add("Chuva");
      System.out.println(c2);
      Collections.sort(c2);
      System.out.println(c2);
      c2.remove("Frio");
      c2.remove(0);
      System.out.println(c2);

   }
}
```

a) Modify it freely to test some of the functions that are available in this class (add, contains, indexOf, lastIndexOf, set, subList, ...).

b) Based on this example, create a new collection (c3) that uses a HashSet, instead of ArrayList, and that contains elements of type Person.

```java
Set<Pessoa> c3 = new HashSet<>();
...
```

Insert 5 different elements and use an iterator to list all the elements on the screen. Check the order of the listing against the order of insertion.
Test the insertion of repeated elements (which cannot exist in a Set).

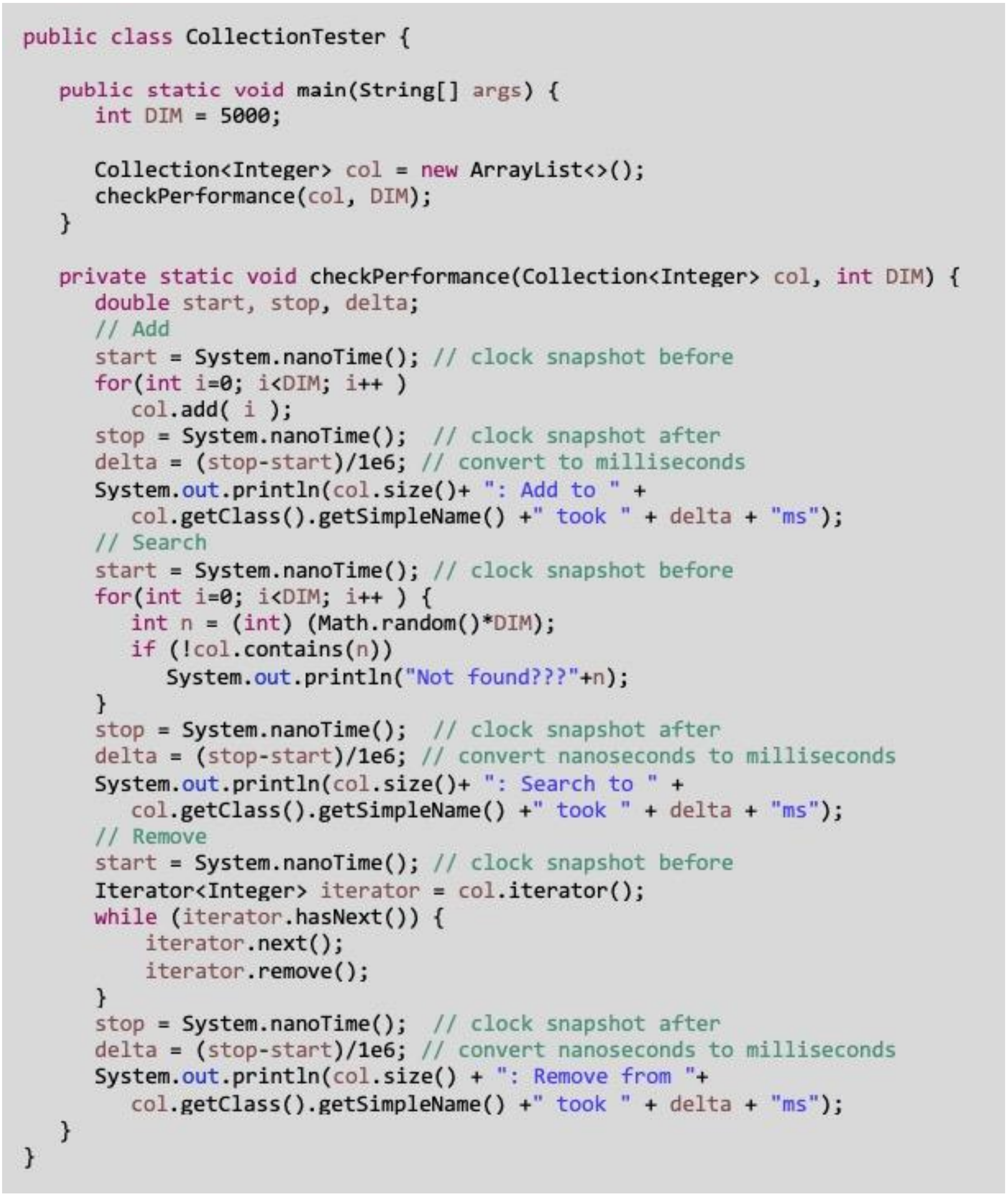
c) Create a new collection (c4) that uses a Date TreeSet (problem 6.1).

```java
Set<Data> c4 = new TreeSet<>();
...
```

Insert 5 different elements and check the order of the list in relation to the insertion order.

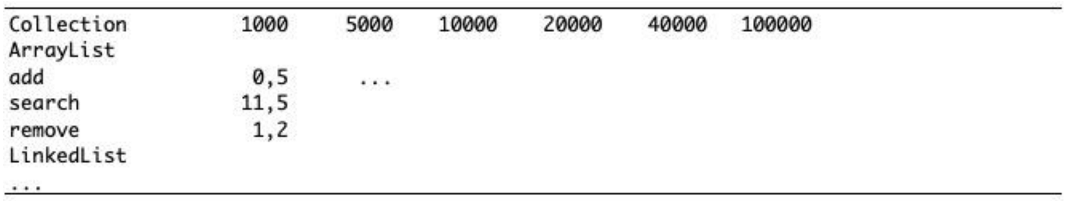
9.2

Using the following code as a base (CollectionTester.java) compare the performance of some of the collections in Java, such as ArrayList, LinkedList, HashSet and TreeSet.



```java
public class CollectionTester {

   public static void main(String[] args) {
      int DIM = 5000;

      Collection<Integer> col = new ArrayList<>();
      checkPerformance(col, DIM);
   }

   private static void checkPerformance(Collection<Integer> col, int DIM) {
      double start, stop, delta;
      // Add
      start = System.nanoTime(); // clock snapshot before
      for(int i=0; i<DIM; i++ )
         col.add( i );
      stop = System.nanoTime();  // clock snapshot after
      delta = (stop-start)/1e6; // convert to milliseconds
      System.out.println(col.size()+ ": Add to " +
         col.getClass().getSimpleName() +" took " + delta + "ms");
      // Search
      start = System.nanoTime(); // clock snapshot before
      for(int i=0; i<DIM; i++ ) {
         int n = (int) (Math.random()*DIM);
         if (!col.contains(n))
            System.out.println("Not found???"+n);
      }
      stop = System.nanoTime();  // clock snapshot after
      delta = (stop-start)/1e6; // convert nanoseconds to milliseconds
      System.out.println(col.size()+ ": Search to " +
         col.getClass().getSimpleName() +" took " + delta + "ms");
      // Remove
      start = System.nanoTime(); // clock snapshot before
      Iterator<Integer> iterator = col.iterator();
      while (iterator.hasNext()) {
          iterator.next();
          iterator.remove();
      }
      stop = System.nanoTime();  // clock snapshot after
      delta = (stop-start)/1e6; // convert nanoseconds to milliseconds
      System.out.println(col.size() + ": Remove from "+
         col.getClass().getSimpleName() +" took " + delta + "ms");
   }
}
```

Adapt the program to measure the results for various dimensions of the collection (for example, creating a table similar to the following). Analyze the results by comparing structures and operations.



```
Collection          1000     5000    10000    20000    40000   100000
ArrayList
add                  0,5      ...
search              11,5
remove               1,2
LinkedList
...
```

Suggestions: modify the checkPerformance method to return the 3 measurements (add, search and remove) and remove all println instructions within this method.

```
private static double[] checkPerformance(Collection<Integer> col, int DIM) {
...
```

Aula10

10.1

Write a program that allows you to save a set of terms (e.g., "white") as well as the meaning of each term (e.g., "That has the color of snow."):
a) Define the data structure (s) that favor the insertion and search speed, and test using a minimum of 5 terms, with the add, change and remove operations.
b) Print the structure, using toString (), listing all the term-meaning pairs, then just the terms and then just the meanings.

10.2

Create a new version of the previous program to ensure that the terms are always ordered and to allow you to store more than one meaning for each term (e.g. "What has the color of snow", "Discolored, pale.", "Color of milk", ...):
a) Repeat the points of the previous problem.
b) Implement a method that, given a term, randomly selects and returns one of its meanings. Suggestion: pass as arguments of this method a reference to the structure and a reference to the search term.

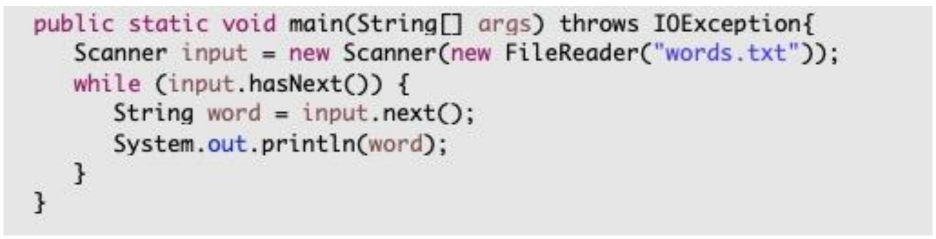
10.3

Write a program that identifies which characters, and at what positions, occur in a String. For example, given the phrase “Hello World” the result should be (you don't need to respect the output format):

```
{ =[5], r=[8], d=[10], e=[1], W=[6], H=[0], l=[2, 3, 9], o=[4, 7]}
```

10.4

The following code excerpt allows you to read all the words (word by word) from a text file, which must be located in the project folder (Eclipse / NetBeans). You can create this file with a text editor or use any java code file.



```
public static void main(String[] args) throws IOException{
   Scanner input = new Scanner(new FileReader("words.txt"));
   while (input.hasNext()) {
      String word = input.next();
      System.out.println(word);
   }
}
```

a) Test the code excerpt in order to list the contents of the file.
b) Keep in a suitable data structure all words with more than 2
characters.
c) List all words ending in 's'.
d) Remove from the structure all words that contain characters other than
letters.

Aula11
11.1

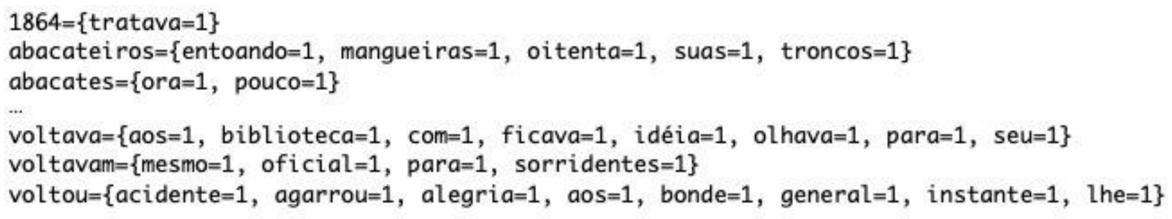
Build a program that reads a text file and that counts all pairs of words found in the file and the number of occurrences of each pair. Disregard all words less than 3 and consider the following characters as separators:

```
\t\n.,:'‘’;?!-*{}=+&/()[]”“\"\'
```

If you use the first sentence of this problem the result in the output file should be as follows:

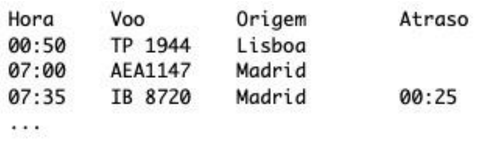
```
cada={par=1}como={separadores=1} considere={como=1} construa={programa=1} conte={todos=1}
despreze={todas=1} encontrados={ficheiro=1} ficheiro={número=1, texto=1} inferior={considere=1} leia={ficheiro=1}
número={ocorrências=1} ocorrências={cada=1} palavras={encontrados=1, tamanho=1} par={despreze=1} pares={palavras=1}
programa={que=1} ...
```

For each word found in the file, the program must associate a set of all the following words, also counting how many times each pair occurs.
Test the program with the file "major.txt". The result should look like:



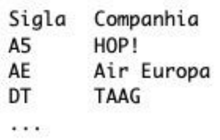
```
1864={tratava=1}
abacateiros={entoando=1, mangueiras=1, oitenta=1, suas=1, troncos=1}
abacates={ora=1, pouco=1}
...
voltava={aos=1, biblioteca=1, com=1, ficava=1, idéia=1, olhava=1, para=1, seu=1}
voltavam={mesmo=1, oficial=1, para=1, sorridentes=1}
voltou={acidente=1, agarrou=1, alegria=1, aos=1, bonde=1, general=1, instante=1, lhe=1}
```

11.2
In the course dossier you will find, in addition to this script, two text files: flights.txt and companies.txt.
The first represents the flights that arrived at Porto airport on May 24th. The structure of this file is as follows (the fields are separated by tab):



```
Hora    Voo       Origem       Atraso
00:50   TP 1944   Lisboa
07:00   AEA1147   Madrid
07:35   IB 8720   Madrid       00:25
...
```

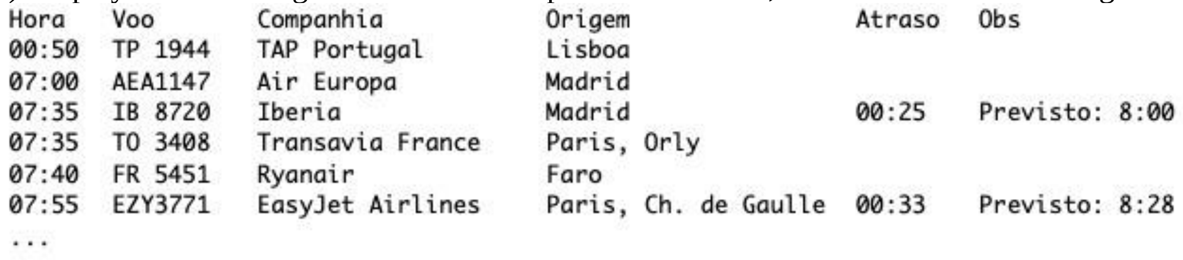
The second contains a table with the acronyms and the names of each company:



```
Sigla  Companhia
A5     HOP!
AE     Air Europa
DT     TAAG
...
```

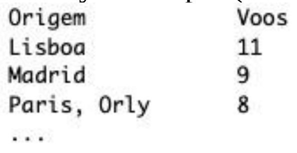
Build a program that reads these two files for suitable structures. Create the Flight class, for example, use sets to store flights in memory, as well as other structures / algorithms that you deem necessary for each of the following points.

a) Display the list of flights with more complete information, as shown in the following table:



```
Hora   Voo      Companhia          Origem                Atraso   Obs
00:50  TP 1944  TAP Portugal       Lisboa
07:00  AEA1147  Air Europa         Madrid
07:35  IB 8720  Iberia             Madrid                00:25    Previsto: 8:00
07:35  TO 3408  Transavia France   Paris, Orly
07:40  FR 5451  Ryanair            Faro
07:55  EZY3771  EasyJet Airlines   Paris, Ch. de Gaulle  00:33    Previsto: 8:28
...
```

b) Save the table in the Infopublico.txt file.
c) Calculate the average of the delays per company and present a table on the screen (Company, Average delay) ordered in ascending order of average delay.
d) Keep in the cities.txt file a table with information such as total arrivals of each home city. Example (sorting by number of flights):



```
Origem         Voos
Lisboa         11
Madrid         9
Paris, Orly    8
...
```

Aula12
12.1

Write a program that reads a text file of your choice and that counts the number of different words in existence. For example, the result could be:

```
  Número Total de Palavras: 161
  Número de Diferentes Palavras: 112
```

12.2

In the course dossier you will find, in addition to this guide, a text file (movies.txt) with the following structure (the fields are separated by tab):

```
name            score     rating      genre          running time
Old School      54.7      R           comedy         92
The Recruit     62.6      PG-13       suspense       115
```

The first line is the header, indicating the order of the attributes of each film; in each of the following lines you will find the name and attributes of each film.

a) Create the Movie class in order to represent each film described in this file.
b) Build a program that links the file to a memory structure (set of Movie) ordered by the name of the movie. List this structure.
c) List the various films on the screen, ordered by: 1) decreasing order of score; 2) by increasing order of "running time".
d) Print the existing genres on the screen, different.
e) Write in the "myselection.txt" file the films with a score higher than 60 and belonging to the comedy genre.