

Considere a existência dos seguintes tipos:

```
public interface Contact{// Contacto
    String getName() // Nome
    LocalDate dateBirth();// Data de nascimento
    String details(); // Dados do contacto
}
```

```
public interface BiConsumer<E1, E2> {
    // Executa a ação sobre e1 e e2
    void accept(E1 e1, E2 e2);
}
```

```
public interface Function<E, R> {
    // Aplica a função a elem.
    R apply(E elem);
}
```

```
public class LocalDate
    implements Comparable<LocalDate>{
    public int getYear(){...}
    public int getMonthValue(){...}
    public int getDayOfMonth(){...}
    public boolean equals(Object o){...}
    // text string such as 2023-01-13
    public String toString(){...}
    public static
        LocalDate parse(String txt){...}
    public static LocalDate now(){...}
    public static
        LocalDate of(int y, int m, int d){...}
    ...
}
```

Grupo I

- a) [3] Realize o método público estático com a seguinte assinatura:

```
<V> void forEachIf( Collection<Contact> noteBook,
                    Function< Contact, V > getvalue,
                    Predicate<V> validate,
                    BiConsumer<Contact, V> action )
```

Que para cada contacto `c` existente em `noteBook` executa a ação binária `action` passando por parâmetro o contacto `c` e o resultado da aplicação da função `getValue` a `c`, enquanto o resultado da aplicação da função `getValue` a `c` cumprir o predicado `validate`.

- b) [3] Utilizando o método da alínea a) e implementando uma `Function` e um `BiConsumer`, realize o método público estático com a seguinte assinatura:

```
SortedSet<Contact> contactsOf( SortedSet<Contact> noteBook,
                               Predicate<Integer> validAge )
```

Que produz e retorna o conjunto de contactos, contidos no conjunto `noteBook`, enquanto a idade no fim do corrente ano cumprir o predicado. O conjunto produzido deve ficar ordenado da mesma forma que o `noteBook` (sobre instâncias de `SortedSet` pode ser chamado o método de instância `comparator()` para obter o `Comparator`).

Nota: Para obter a idade use o método estático `now` da classe `LocalDate` para obter a data corrente, e o método de instância `getYear` para obter o ano.

- c) [3] Realize o método público estático com a seguinte assinatura:

```
List<Contact> contactsInAgeRange (SortedMap<Integer, Set<Contact>> contactsPerAge,
                                  int minAge, int maxAge)
```

Que produza e retorne uma lista com os contactos, existentes no contentor associativo ordenado `contactsPerAge`, cuja idade está no intervalo entre `minAge` e `maxAge`. No contentor associativo as chaves são a idade e o valor associado é o conjunto de contactos que tem essa idade. O contentor associativo está ordenado de forma crescente. A lista deverá ficar ordenada por idades decrescentes e para a mesma idade pela ordem que se encontram no `contactsPerAge`.

Grupo II

- a) [4] Realize o método público estático com a seguinte assinatura:

```
public interface BiConsumer<E1, E2> {  
    void accept(E1 e1, E2 e2);  
}  
public interface Predicate<E> {  
    boolean test(E e);  
}
```

```
int forEachLine(String pathnameIn,  
                Predicate<LocalDate> pred,  
                BiConsumer<String, LocalDate> action) throws IOException
```

Que leia o ficheiro de texto com pathname `pathnameIn`, onde cada linha contém a data de nascimento e o nome de uma pessoa (com o formato <aaaa-mm-dd> <nome da pessoa>), e que execute a ação binária `action` caso a data de nascimento obedeça ao predicado `pred`. Ao método `accept`, evocado sobre `action`, deve ser passado o nome e a data. O método `forEachLine` retorna o número de datas que obedecem ao predicado.

Use o método estático `parse` da classe `LocalDate` que recebendo por parâmetro uma `string` que representa uma data (ex. 2023-01-13), retorna a respetiva `LocalDate`.

```
1998-07-30 Jorge Silva  
2003-01-30 Ana Coelho  
2010-01-30 Pedro Jorge  
1997-01-11 Miguel Gil  
1998-01-30 Vítor Palma
```

datesOfBirth.txt

- b) [2] Utilizando o método da alínea a) e implementado um `Predicate` e um `BiConsumer` realize o método público estático com a seguinte assinatura:

```
void copyBirthdays(String pathnameIn, String pathnameOut) throws IOException
```

Que leia do ficheiro de texto de nome `pathnameIn`, onde cada linha contém a data de nascimento e o nome de uma pessoa, e produza um ficheiro de texto de nome `pathnameOut` contendo o nome das pessoas que fazem anos na data corrente e a respetiva idade. A figura ilustra uma utilização do método pedido para o dia 30 de Janeiro de 2023.

```
Ana Coelho faz 20 anos  
Pedro Jorge faz 13 anos  
Vítor Palma faz 25 anos
```

Nota: Use o método estático `now` da classe `LocalDate` para obter a data corrente e os métodos `getDayOfMonth`, `getMonthValue` e `getYear` para obter o dia, o mês e o ano respetivamente.

- c) [5] Faça uma aplicação com o aspeto da figura. Quando for premido o botão “birthdays” deve ser chamado o método `forEachLine` da alínea a), tendo em conta o conteúdo das caixas de texto. Na área de texto “names” deve ser listado a idade e o nome das pessoas que constam no ficheiro e que fazem anos no dia e mês presentes nas caixas de texto. Em caso de erro no acesso ao ficheiro deve ser chamado o método estático `JOptionPane.showMessageDialog` passando como 1º parâmetro a própria janela e como 2º parâmetro a `string` “Error:” seguida da mensagem que especifica o erro.

Interface usada na receção de eventos de ação:

```
public interface ActionListener {  
    void actionPerformed(ActionEvent e);  
}
```

Para a instanciação das caixas e da área de texto use os métodos:

```
public static JTextField newJTextField( String title, int dim) {  
    JTextField tf = new JTextField( dim );  
    tf.setBorder( new TitledBorder( title ) ); return tf;  
}  
public static JTextArea newJTextArea( String title) {  
    JTextArea ta = new JTextArea( 10, 35 );  
    ta.setBorder( new TitledBorder( title ) ); return ta;  
}
```

