052510 - Ingegneria del Software

Esercitazione 5: Programmazione funzionale

Federico Giannini

federico.giannini@polimi.it

Esercizio 1 – Esame 16/01/2018

Si consideri la classe `Couple` così definita:

```
public class Couple {
    public int val1;
    public int val2;
}
```

e una lista `data` di istanze di tale classe, opportunamente inizializzata:

```
List<Couple> data = new ArrayList<Couple>();
data.add(...); ...; data.add(...);
```

Si scriva un frammento di programma Java 8 che usa lo stile funzionale per restituire la somma di tutti i secondi elementi (`val2`) delle coppie nella lista `data` il cui primo elemento sia maggiore o uguale a zero.

Esercizio 2 – Esame 02/09/2016

Trasformare il seguente codice Java in stile funzionale.

```
public static void numeroPari(final List<Integer> numeri) {

  Integer found = null;
  for (Integer i: numeri) {
    if (i % 2 ==0) {
      found = i;
      break;
    }
  }
  if (found != null) {
    System.out.print("Il primo numero pari " + found);
  } else {
    System.out.println("Nessun numero pari!");
  }
}
```

Esercizio 3 – Esame 26/06/2019

Si consideri il metodo `findMyIntegers`, non completamente definito, della seguente classe `Exam`:

```
public class Exam {
   public static List<Integer> findMyIntegers(List<Integer> aList, Predicate<Integer> aPredicate) {
      BiFunction<List<Integer>, Predicate<Integer>, List<Integer>> onlyMine;
      onlyMine = (myListOfInt, myPredicate) -> {
         ..........................................
         ..........................................
      };
      return onlyMine.apply(aList, aPredicate);
   }

   public static void main(String[] s){
      Predicate<Integer> predicate = x -> x % 2 == 0;
      List<Integer> startingList = Arrays.asList(1, 7, 4, 6, 8, 9);
      List<Integer> finalList = findMyIntegers(startingList, predicate);
      System.out.println(finalList);
   }
}
```

Il metodo applica il predicato `aPredicate`, definito con l'interfaccia funzionale `Predicate<Integer>`, al parametro `aList`, restituendone, come una nuova lista, tutti e soli i valori che soddisfano il predicato stesso.

A tale scopo, il codice utilizza l'interfaccia funzionale `Bifunction` per definire una funzione che riceve come argomenti una lista e un `Predicate` di `Integer`, restitutendo una lista.

Esercizio 3 – Esame 26/06/2019

La definizione delle due interfacce funzionali viene ricordata di seguito:

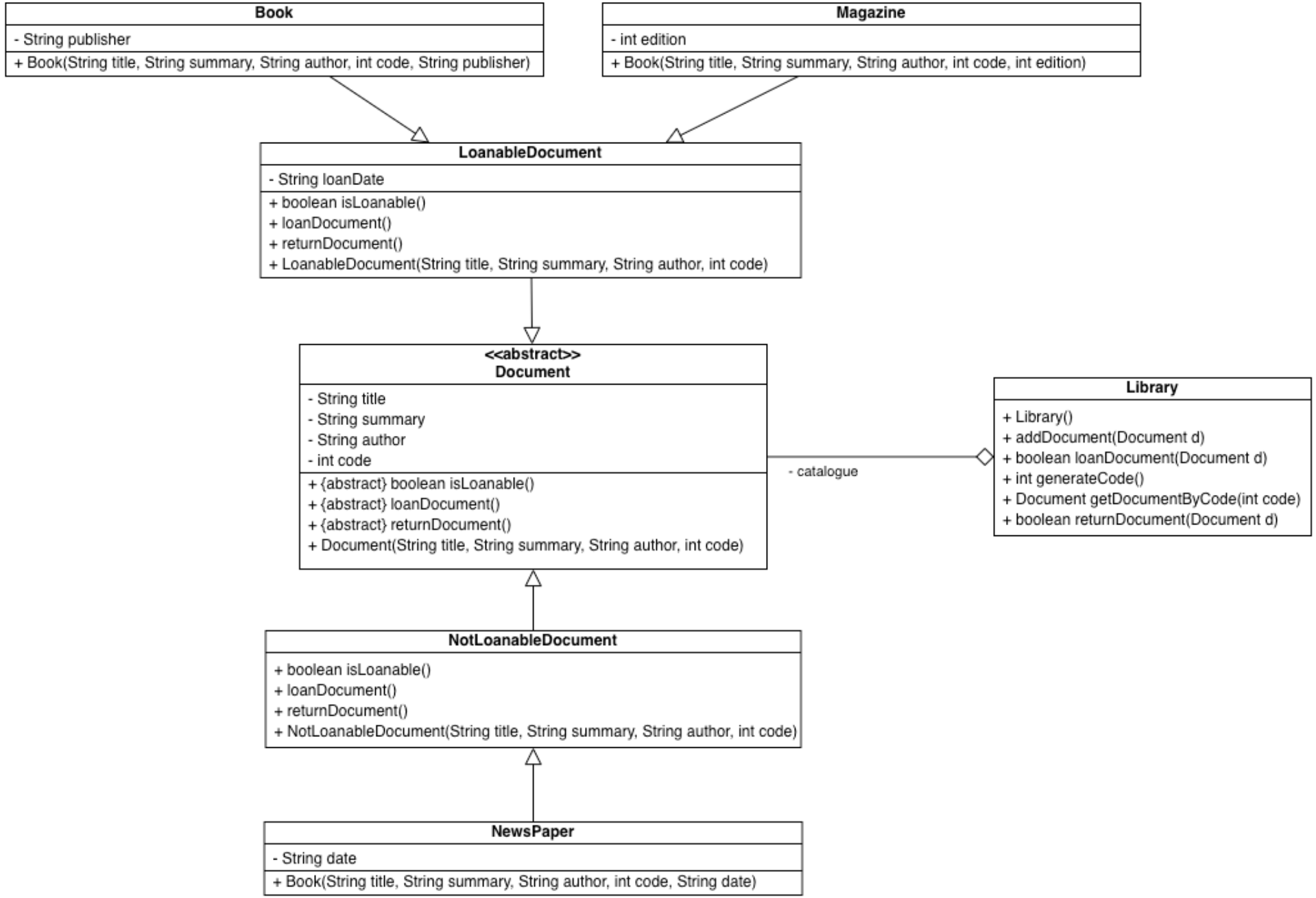
```
@FunctionalInterface
public interface BiFunction<T,U,R> {
   // Apply this function to two arguments;
   R apply(T t, U u);
}

@FunctionalInterface
public interface Predicate<T> {
   // Evaluates this predicate on the given argument:
   boolean test(T t);
   ...
}
```

Completare il codice del metodo usando preferibilmente un approccio funzionale. Indicare inoltre il risultato dell'esecuzione del metodo `main` interno alla classe `Exam`.

Esercizio 4 – Si ritorna a programmare ☺

- Si vuole realizzare un sistema per gestire il catalogo di una Biblioteca.
- Il sistema deve poter gestire diverse categorie di documenti: libri, riviste e giornali, ciascuno con caratteristiche diverse
- Libri e riviste possono essere prestati, giornali e altri documenti possono solo essere consultati.
- Il sistema deve poter stampare un catalogo dettagliato dei documenti con una breve sintesi del contenuto.
- Produrre l'UML Class Diagram e implementare la soluzione in Java.


Book
- String publisher
+ Book(String title, String summary, String author, int code, String publisher)
Magazine
- int edition
+ Book(String title, String summary, String author, int code, int edition)
LoanableDocument
- String loanDate
+ boolean isLoanable()
+ loanDocument()
+ returnDocument()
+ LoanableDocument(String title, String summary, String author, int code)
<<abstract>>
Document
- String title
- String summary
- String author
- int code
+ {abstract} boolean isLoanable()
+ {abstract} loanDocument()
+ {abstract} returnDocument()
+ Document(String title, String summary, String author, int code)
- catalogue
Library
+ Library()
+ addDocument(Document d)
+ boolean loanDocument(Document d)
+ int generateCode()
+ Document getDocumentByCode(int code)
+ boolean returnDocument(Document d)
NotLoanableDocument
+ boolean isLoanable()
+ loanDocument()
+ returnDocument()
+ NotLoanableDocument(String title, String summary, String author, int code)
NewsPaper
- String date
+ Book(String title, String summary, String author, int code, String date)