

Lösningsförslag, omkontrollskrivning PTDC

2012-04-11

```
1. public class BorrowObject {
    private Person person;
    private Book book;
    private String borrowDate, lastReturnDate, returnedDate;

    public BorrowObject(Person person, Book book) {
        this.book = book;
        this.person = person;
        borrowDate = DateKeeper.today();
        if (book.isReserved()) {
            lastReturnDate = DateKeeper.getDate(10);
        } else {
            lastReturnDate = DateKeeper.getDate(21);
        }
        returnedDate = "Not returned";
    }

    public Book getBook() {
        return book;
    }

    public Person getPerson() {
        return person;
    }

    public String getBorrowDate() {
        return borrowDate;
    }

    public String getLastReturnDate() {
        return lastReturnDate;
    }

    public String getReturnedDate() {
        return returnedDate;
    }

    public void setReturned() {
        returnedDate = DateKeeper.today();
    }
}
```

```
2. public class Register {
    private ArrayList<Book> books;
    private ArrayList<BorrowObject> current;
    private ArrayList<BorrowObject> past;

    public Register() {
        books = new ArrayList<Book>();
        current = new ArrayList<BorrowObject>();
        past = new ArrayList<BorrowObject>();
    }

    private Book findBook(int bookId) {
        for (int i = 0; i < books.size(); i++) {
            if (books.get(i).getId() == bookId) {
                return books.get(i);
            }
        }
        return null;
    }

    public void borrowedBook(int bookId, Person person) {
        Book b = findBook(bookId);
        current.add(new BorrowObject(person, b));
    }

    public int returnedBook(int bookId, int libraryId) {
        int i = 0;
        while (i < current.size() && current.get(i).getBook().getId() != bookId) {
            i++;
        }
        if (i == current.size()) {
            return -1;
        }
        BorrowObject bo = current.get(i);
        bo.setReturned();
        current.remove(i);
        past.add(bo);
        if (bo.getBook().getLibraryId() == libraryId) {
            return 0;
        } else {
            return bo.getBook().getLibraryId();
        }
    }

    public ArrayList<BorrowObject> getWarning(int days) {
        ArrayList<BorrowObject> warned = new ArrayList<BorrowObject>();
        for (int i = 0; i < current.size(); i++) {
            if (current.get(i).getLastReturnDate().equals(
                DateKeeper.getDate(days))) {
                warned.add(current.get(i));
            }
        }
        return warned;
    }
}
```

```

3. public class TimeTest {
    public static void main(String[] args) {
        System.out.print("Mata in det totala antalet tal och antalet tal att ta bort: ");
        Scanner scan = new Scanner(System.in);
        int totalSize = scan.nextInt();
        int partSize = scan.nextInt();
        TestCase tc = new TestCase(totalSize, partSize);
        int[] all = tc.getAllNumbers();
        int[] remove = tc.getPartNumbers();

        long start = System.currentTimeMillis();
        int[] ar = new int[all.length];
        for (int i = 0; i < all.length; i++) {
            ar[i] = all[i];
        }
        int arSize = ar.length;
        for (int i = 0; i < remove.length; i++) {
            int j = 0;
            while (ar[j] != remove[i]) {
                j++;
            }
            for (int pos = j; pos < arSize - 1; pos++) {
                ar[pos] = ar[pos + 1];
            }
            arSize--;
        }
        System.out.println("Testtime Array: "
            + (System.currentTimeMillis() - start));

        start = System.currentTimeMillis();
        ArrayList<Integer> al = new ArrayList<Integer>();
        for (int i = 0; i < all.length; i++) {
            al.add(new Integer(all[i]));
        }
        for (int i = 0; i < remove.length; i++) {
            int j = 0;
            while (al.get(j).intValue() != remove[i]) {
                j++;
            }
            al.remove(j);
        }
        System.out.println("Testtime ArrayList: "
            + (System.currentTimeMillis() - start));
    }
}

4. int[] allNumbers = new int[totalSize];
Random rand = new Random();
HashSet<Integer> created = new HashSet<Integer>();
for (int i = 0; i < totalSize; i++) {
    int randNbr = 0;
    do {
        randNbr = rand.nextInt(Integer.MAX_VALUE);
    } while (created.contains(randNbr));
    created.add(randNbr);
    allNumbers[i] = randNbr;
}

```
