COMP 131 Midterm Exam II, Practice Exam (100 points total)

name: SOLUTION

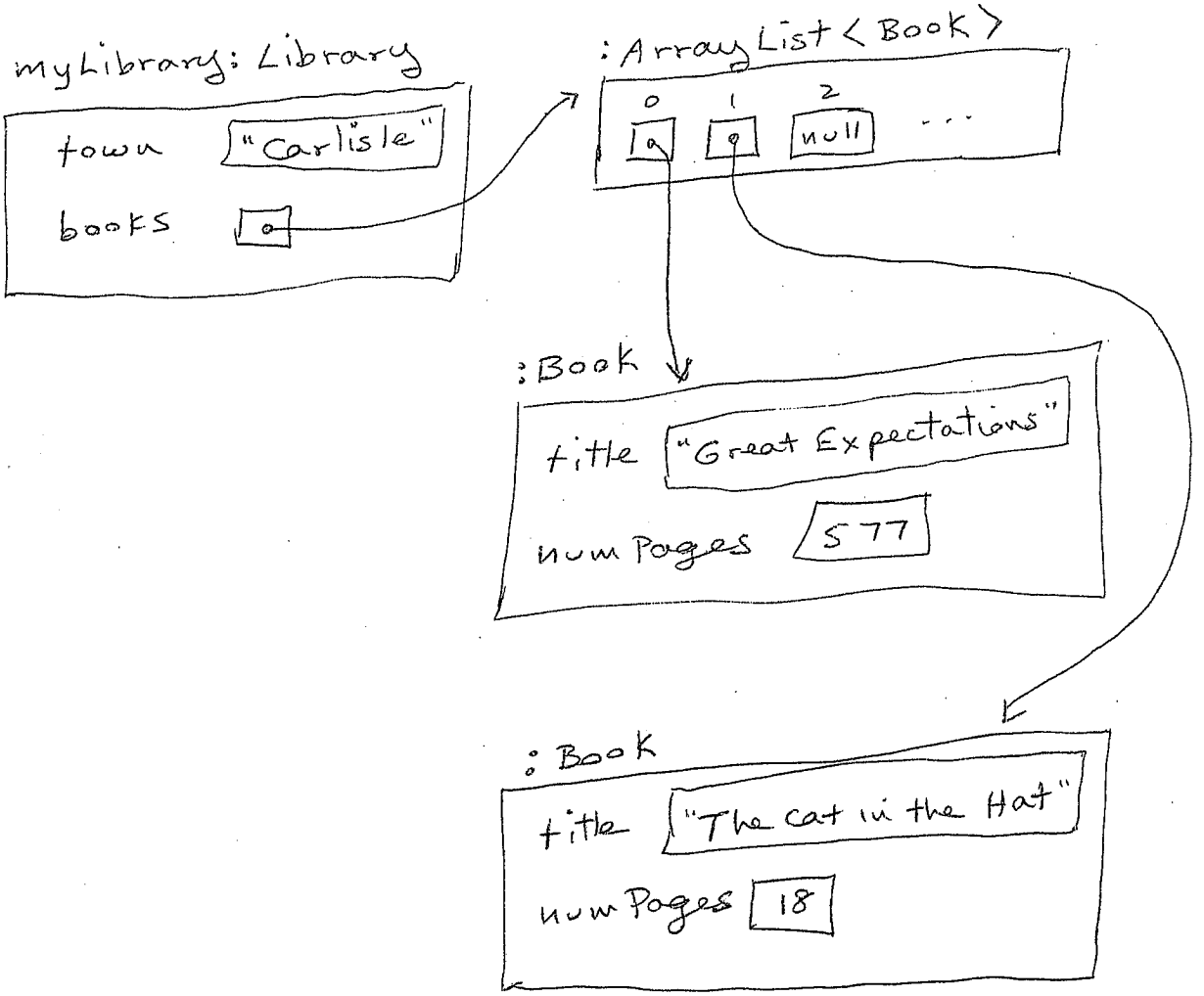
Question 1 (15 points). Consider the following class definitions (JavaDoc comments have been eliminated to save space):

```
public class Book {
      private String title; // the book's title
      private int numPages; // the number of pages in the book

      public Book(String title, int numPages) {
            this.title = title;
            this.numPages = numPages;
      }

      public String getTitle() {
            return title;
      }

      public int getNumPages() {
            return numPages;
      }
}

import java.util.ArrayList;

public class Library {
      private String town; // the name of the town
                           // where this library is located
      private ArrayList<Book> books; // list of all books
                                     // contained in the library

      public Library(String town) {
            this.town = town;
            books = new ArrayList<Book>();
      }

      public void addBook(Book book) {
            books.add(book);
      }

      public void removeBook(int index) {
            books.remove(index);
      }
}
```

Draw an object diagram for the object referred to by the variable `myLibrary` after the following snippet of code has been executed.

(Question 1 continued)

```
Library myLibrary = new Library("Carlisle");
Book greatExpectations =
     new Book("Great Expectations", 577);
Book prideAndPrejudice =
     new Book("Pride and Prejudice", 322);
Book catInHat = new Book("The Cat in the Hat", 18);
Book websters = new Book("Webster's Dictionary", 759);

myLibrary.addBook(greatExpectations);
myLibrary.addBook(prideAndPrejudice);
myLibrary.addBook(catInHat);
myLibrary.addBook(websters);
myLibrary.removeBook(3);
myLibrary.removeBook(1);
```



Question 2 (10 points). Rewrite the `removeBook` method so that it prints an appropriate error message if the formal parameter `index` is too large or too small, but still removes the correct book if `index` is valid.

```
public void removeBook (int index){
    if (index >=0 && index < books.size()){
        books.remove(index);
    }
    else {
        System.out.println("invalid index");
    }
}
```

Question 3 (15 points). Write a method for the `Library` class with the signature `public Book getLongestBook()`, which returns the longest book in the library (i.e. the one with the most pages). If there is a tie for the longest book, the method may return any one of the longest books.

```
public Book getLongestBook() {
    if (books.size() == 0){
        return null;
    }
    else {
        Book longest = books.get(0);
        for (int index=0; index < books.size(); index++){
            Book aBook = books.get(index);
            if (aBook.getNumPages() >
                    longest.getNumPages()) {
                longest = aBook;
            }
        }
        return longest;
    }
}
```

Question 4 (10 points). Write a method for the `Library` class with the signature `public void printLongestBook()`, which prints out the title of the longest book in the library. Hint: for maximum credit, do not repeat or rewrite any code you have written already in the previous questions.

```
public void printLongestBook() {
    Book longest = getLongestBook();
    if (longest != null) {
        System.out.println(longest.getTitle());
    }
    else {
        System.out.println("no books in the library.");
    }
}
```

Question 5 (15 points). Write a method for the `Library` class with the signature `public Library getBooksOver500()`, which returns a new library containing all books in the original library with 500 or more pages.

```
public Library getBooksOver500() {
    Library answer = new Library("carlisle");
    for (int index=0; index < books.size(); index++) {
        if (books.get(index).getNumPages() >= 500) {
            answer.add(books.get(index));
        }
    }
    return answer;
}
```

Question 6 (5 points). Write a snippet of Java code that creates a local variable named `greatGatsby`, and stores in that variable a reference to a new `Book` object representing a 233-page book whose title is "The Great Gatsby".

```
Book greatGatsby = new Book("The Great Gatsby",
                            233);
```

Question 7 (8 points). Fill in the two missing Boolean expressions in the if statements of the following method of the Book class.

```
/**
 * Return true if book2 has the same title and
 * number of pages as the calling object,
 * otherwise return false.
 */
public boolean equals(Book book2) {
      String title1 = this.getTitle();
      String title2 = book2.getTitle();
      int numPages1 = this.getNumPages();
      int numPages2 = book2.getNumPages();

      //  first test whether the two titles are the same
      if ( title1.equals(title2) ) {
            // next test if numbers of pages are the same
            if ( numPages1 == numPages2 ) {
                  return true;
            } else {
                  return false;
            }
      } else {
            return false;
      }
}
```

Question 8 (10 points). One definition of *composition* is "using objects as fields in other objects". Write 3-4 sentences explaining the advantages of using composition in software projects.

Composition allows us to build complex software from smaller simpler parts. This is advantageous because it allows different people to work on different parts of the code at the same time, and it means that developers do not need to understand the entire project or think about all of the complexity at once.

Question 9 (6 points). What output would be produced by the following snippet of code?

```
int value = 29;
while (value >= 14) {
      System.out.println(value);
      value = value - 5;
}
```

29
24
19
14

Question 10 (6 points). What output would be produced by the following snippet of code?

```
for (int i = 5; i < 25; i = 2 * i) {
      int j = i - 3;
      System.out.println(j);
}
```

2
7
17