SWEN20003
Object Oriented Software Development
Workshop 2 (Solutions)

Eleanor McMurtry

Semester 2, 2020

1. Create a `Circle` class with a *radius*, *x coordinate*, and *y coordinate*.
   (a) Add a default constructor `public Circle()` that sets the radius to `1` and the coordinates to `(0, 0)`.
   (b) Add a constructor `public Circle(double radius)` that sets the radius to the argument value, and the coordinates to `(0, 0)`.
   (c) Add a constructor `public Circle(double radius, double x, double y)` with the appropriate actions.
   (d) Add `toString` and `equals` methods.

   Solution:

```
public class Circle {
    private double radius;
    private double x;
    private double y;

    public Circle() {
        this.radius = 1;
        this.x = 0;
        this.y = 0;
    }

    public Circle(double radius) {
        this.radius = radius;
        this.x = 0;
        this.y = 0;
    }

    public Circle(double radius, double x, double y) {
        this.radius = radius;
        this.x = x;
        this.y = y;
    }

    public String toString() {
        return String.format("Circle at (%f, %f) with radius %f", x, y, radius);
    }

    public boolean equals(Circle rhs) {
        double EPSILON = 1e-5;
        return Math.abs(radius - rhs.radius) < EPSILON
                && Math.abs(x - rhs.x) < EPSILON
                && Math.abs(y - rhs.y) < EPSILON;
    }
}
```

2. Create a similar `Rectangle` class with a *left coordinate*, a *top coordinate*, a *width*, and a *height*. **Solution:**

```
public class Rectangle {
    private double left;
    private double top;
    private double width;
    private double height;

    public Rectangle() {
        this.left = 0;
        this.top = 0;
        this.width = 1;
        this.height = 1;
    }

    // etc.

    public String toString() {
        return String.format("Rectangle from (%f, %f) to (%f, %f)", left, top, left + width, top +
    }

    public boolean equals(Rectangle rhs) {
        double EPSILON = 1e-5;
        return Math.abs(left - rhs.left) < EPSILON
                && Math.abs(top - rhs.top) < EPSILON
                && Math.abs(width - rhs.width) < EPSILON
                && Math.abs(height - rhs.height) < EPSILON;
    }
}
```

3. (a) Create a `Book` class to represent a book in a library. Books have an *author*, a *title*, and can either be *borrowed* or *not borrowed*.
   (b) Write a constructor for your class.
   (c) Define getters for your class.
   (d) Add appropriate `toString` and `equals` methods to your class. **Solution:**

```
public class Book {
    private String author;
    private String title;
    private boolean borrowed = false;

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

    public String getAuthor() {
        return author;
    }

    public String getTitle() {
        return title;
    }

    public boolean isBorrowed() {
        return borrowed;
    }

    public String toString() {
        return author + ": " + title;
    }
```

```
    public boolean equals(Book rhs) {
        return author.equals(rhs.author) && title.equals(rhs.title);
    }
}
```

(e) Define a method `void borrow(String borrowedBy)` that marks the book as *borrowed*. You'll need to add an attribute to the class to store who has borrowed the book.

(f) Define a method `void returnBook()` that returns the book to the library.

(g) Add a static attribute to count the number of books that are currently borrowed.

(h) Define a static method that returns the number of books currently borrowed. **Solution:**

```
private String borrowedBy = null;
private boolean borrowed = false;

public void borrow(String borrowedBy) {
    if (!borrowed) {
        borrowed = true;
        this.borrowedBy = borrowedBy;

        ++Book.numBorrowed;
    }
}

public void returnBook() {
    if (borrowed) {
        borrowed = false;
        borrowedBy = null;
        --Book.numBorrowed;
    }
}

public static int numBorrowed() {
    return numBorrowed;
}
```

4. (a) Create a `Library` class with an appropriate constructor to represent a library that can hold up to `10` books. (Hint: use an array!)

(b) Define a method to add a book to the library. If the library is already full, it should do nothing.

(c) Define a method `Book lookup(String title)` that looks up a book by title and returns the first book with that title in the library. If there is no such book, it should return `null`.

(d) Add an overloaded method `Book lookup(String title, String author)` that looks up a book by title *and* author.

(e) Add a method `String getCatalogue()` that returns a string containing each book in the library on separate lines, in the following format:

```
Charles Dickens: Great Expectations
Sun Tzu: The Art of War
Brian Kernighan & Denis Ritchie: The C Programming Language
```

(Hint: if you define `Book`'s `toString` method carefully, this problem is easy.) **Solution:**

```
public class Library {
    private static final int MAX_BOOKS = 10;
    private int numBooks = 0;
    private Book[] books;

    public Library() {
        this.books = new Book[10];
    }
```

```
    public void addBook(Book book) {
        if (numBooks < MAX_BOOKS) {
            books[numBooks] = book;
            ++numBooks;
        }
    }

    public Book lookup(String title) {
        for (int i = 0; i < numBooks; ++i) {
            if (books[i].getTitle().equals(title)) {
                return books[i];
            }
        }

        return null;
    }

    public Book lookup(String title, String author) {
        for (int i = 0; i < numBooks; ++i) {
            Book book = books[i];
            if (book.getTitle().equals(title) && book.getAuthor().equals(author)) {
                return book;
            }
        }

        return null;
    }

    public String getCatalogue() {
        String result = "";
        for (int i = 0; i < numBooks; ++i) {
            result += books[i].toString() + "\n";
        }
        return result;
    }
}
```

(f) Replace the static attribute and method in the `Book` class with an instance variable and method in the `Library` class. **Solution:**

```
public int numBorrowed() {
    int count = 0;
    for (int i = 0; i < numBooks; ++i) {
        if (books[i].isBorrowed()) {
            ++count;
        }
    }
    return count;
}
```

(g) Write a `main` method to create some books, add them to a library, look up books, and borrow them. **Solution:**

```
public static void main(String[] args) {
    Library library = new Library();
    library.addBook(new Book("Charles Dickens", "Great Expectations"));
    library.addBook(new Book("Sun Tzu", "The Art of War"));
    library.addBook(new Book("Brian Kernighan & Denis Ritchie", "The C Programming Language"));
    System.out.print(library.getCatalogue());

    Book dickens = library.lookup("Great Expectations");
    dickens.borrow("Eleanor McMurtry");
```

```
        System.out.println("Number of books borrowed: " + library.numBorrowed());
    }
```