EXAMPLE – BOOK OBJECT (INSTANCE METHODS AND FIELDS)

zyBook 9.2, zyBook 9.3, zyBook 9.4, zyBook 9.10
Oracle - Java Tutorial: Controlling Access to Members of a Class
Oracle - Java Tutorial: Using the this Keyword
Oracle - Java Tutorial: Classes

```java
import java.util.Date;

/**
 * An class representing a book
 * @author Jessica Young Schmidt
 */
public class Book {
    /** Constant with number of milliseconds in a day */
    private static final long ONE_DAY = 86400000;

    /** Standard length of checkout in days */
    private static final int CHECKOUT_LENGTH = 90;

    /** Title of the book */
    private String title;

    /** Author of the book */
    private String author;

    /** Publication year of the book */
    private int pubYear;

    /** Person who has checked out the book */
    private String checkedOut;

    /** Location of the book in the stacks */
    private int location;

    /** Date a checked out book is due */
    private Date dueDate;
```

Full code available at: https://go.ncsu.edu/example-objects-instance-methods-fields

```java
/**
 * Returns the book title
 * @return book title
 */
public String getTitle() {
    return title;
}
/**
 * Sets book title to the given parameter
 * @param title new book title
 */
public void setTitle(String title) {
    this.title = title;
}

/**
 * Returns the book author
 * @return book author
 */
public String getAuthor() {
    return author;
}

/**
 * Sets book author to the given parameter
 * @param author new book author
 */
public void setAuthor(String author) {
    this.author = author;
}
```

```java
/**
 * Returns publication year
 * @return publication year
 */
public int getPubYear() {
    return pubYear;
}

/**
 * Sets book title to the given parameter
 * @param pubYear new publication year
 */
public void setPubYear(int pubYear) {
    this.pubYear = pubYear;
}

/**
 * Returns the location of Book
 * @return location of Book
 */
public int getLocation() {
    return location;
}

/**
 * Sets location of Book based on parameter
 * @param location new location of this Book
 */
public void setLocation(int location) {
    this.location = location;
}
```

```
    /**
     * Return unity ID for person who has checked out the book. Returns null if not checked out.
     * @return the unity ID for person who has checked out the book
     */
    public String getCheckedOut() {
        return checkedOut;
    }

    /**
     * Checks out a book if not already checked out
     * @param unityID unityID to check book out to
     * @return true if able to check out to the unity id parameter. false if book already checked out
     */
    public boolean checkOut(String unityID) {
        if (checkedOut == null) {
            checkedOut = unityID;
            dueDate = new Date(System.currentTimeMillis() + CHECKOUT_LENGTH * ONE_DAY);
            return true;
        }
        return false;
    }

    /**
     * Checks in a book
     */
    public void checkIn() {
        checkedOut = null;
        dueDate = null;
    }

    /**
     * Returns the due date for the book, or null if book is not checked out
     * @return the due date for book.
     */
    public Date getDueDate() {
        return dueDate;
    }
}
```

Client Code

```java
import java.util.Scanner;
public class HuntLibrary {
  public static int NUM_BOOKS = 1000;
  public static void main(String[] args) {
    // Parallel arrays for all book items
    String[] authors = new String[NUM_BOOKS];
    String[] titles = new String[NUM_BOOKS];
    int[] pubYear = new int[NUM_BOOKS];
    Scanner input = new Scanner(System.in);
    for (int i = 0; i < authors.length; i++) {
      System.out.print("Author: ");
      authors[i] = input.nextLine();
      System.out.print("Title: ");
      titles[i] = input.nextLine();
      System.out.print("Publication year: ");
      pubYear[i] = input.nextInt();
      input.nextLine(); //To clear line after int
    }
  }
}
```

```java
import java.util.Scanner;
public class HuntLibrary {
  public static int NUM_BOOKS = 1000;
  public static void main(String[] args) {
    Book[] books = new Book[NUM_BOOKS];
    Scanner input = new Scanner(System.in);
    for (int i = 0; i < books.length; i++) {
      books[i] = new Book();
      System.out.print("Author: ");
      books[i].setAuthor(input.nextLine());
      System.out.print("Title: ");
      books[i].setTitle(input.nextLine());
      System.out.print("Publication year: ");
      books[i].setPubYear(input.nextInt());
      input.nextLine(); //To clear line after int
    }
  }
}
```

Unit and Integration Test

```
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

/**
 * Testing Book Class
 *
 * @author Jessica Young Schmidt
 */
public class BookTest {
    /** Book that is used for testing */
    private Book rosie;

    /**
     * Set up field that will be used in each test. Method is run before each test
     * method.
     */
    @BeforeEach
    public void setup() {
        rosie = new Book();
    }

    /**
     * Tests author methods
     */
    @Test
    public void testAuthor() {
        assertNull(rosie.getAuthor(), "Check that author is initially null");
        rosie.setAuthor("Andrea Beaty");
        assertEquals("Andrea Beaty", rosie.getAuthor(), "Check that author was updated");
    }
```

```
/**
 * Tests title methods
 */
@Test
public void testTitle() {
    assertNull(rosie.getTitle(), "Check that title is initially null");
    rosie.setTitle("Rosie Revere, Engineer");
    assertEquals("Rosie Revere, Engineer", rosie.getTitle(), "Check that title was updated");
}

/**
 * Tests publication year methods
 */
@Test
public void testPubYear() {
    assertEquals(0, rosie.getPubYear(), "Check that pubYear is initially 0");
    rosie.setPubYear(2013);
    assertEquals(2013, rosie.getPubYear(), "Check that pubYear was updated");
}

/**
 * Tests location methods
 */
@Test
public void testLocation() {
    assertEquals(0, rosie.getLocation(), "Check that location is initially 0");
    rosie.setLocation(116);
    assertEquals(116, rosie.getLocation(), "Check that location was updated");
}
```

```
    /**
     * Tests methods for checking book out and in
     */
    @Test
    public void testCheckOutIn() {
        assertNull(rosie.getCheckedOut(), "Check that checkedOut is initially null");
        assertNull(rosie.getDueDate(), "Check that dueDate is initially null");
        assertTrue(rosie.checkOut("jdyoung2"), "Check that can check out the book");
        assertEquals("jdyoung2", rosie.getCheckedOut(), "Check that checked out to jdyoung2");
        assertNotNull(rosie.getDueDate(), "Check that dueDate is not null");
        assertFalse(rosie.checkOut("jdoe"), "Check that cannot check out the book");
        assertEquals("jdyoung2", rosie.getCheckedOut(), "Check that checked out to jdyoung2");
        assertNotNull(rosie.getDueDate(), "Check that dueDate is not null");
        rosie.checkIn();
        assertNull(rosie.getCheckedOut(), "Check that checkedOut is null after book is returned");
        assertNull(rosie.getDueDate(), "Check that dueDate is null after book is returned");
        assertTrue(rosie.checkOut("jdoe"), "Check that can check out the book");
        assertEquals("jdoe", rosie.getCheckedOut(), "Check that checked out to jdoe");
        assertNotNull(rosie.getDueDate(), "Check that dueDate is not null");
    }

}
```