**44-542 Object Oriented Programming Exam 02 Part 02 (40 Points) KEY**

**Do NOT use lambda expressions on this exam.**

All of the questions on this part of the exam refer to the classes on the Books handout. If you do not have this handout, ask your instructor for a copy.

1. (10 pts) Note that **IllegalISBNException** extends **RuntimeException** and therefore is an unchecked exception and does not have to be advertised.

In the **BookList** class, rewrite method **addBook** so that an **IllegalISBNException** is thrown if the length of the ISBN of the book passed as an argument is not equal to 10. Otherwise, the book is added to the array list **books**. The message **Bad ISBN!** should be included as an argument to the exception.

Write your answer in the space below. Write the complete method, including the method header, but write ***only*** method **addBook**.

**public void addBook(Book book) {**

**if (book.getISBN().length() != 10) {**

**throw new IllegalISBNException("Bad ISBN!");**

**}**

**books.add(book);**

**}**

1. (10 pts) In the **BookDriver** class, add code to the **while** loop to catch the **IllegalISBNException** that you threw in the previous question on this exam.

Example: Suppose **books.txt** has this data

**123456789X 375**

**34573412067 418**

**8889990005 399**

**12312312 83**

The output produced by the driver would be:

**Skipping book with ISBN = 34573412067 Invalid ISBN.**

**books.IllegalISBNException: Bad ISBN!**

**Skipping book with ISBN = 12312312 Invalid ISBN.**

**books.IllegalISBNException: Bad ISBN!**

Your code must work for any valid data set. The output produced must match exactly the output shown above.

Write your answer in the space below. Write ***only*** the **while** loop. ***Do not change the three lines currently in the* while** ***loop***.

**while (dataIn.hasNext()) {**

**try {**

**ISBN = dataIn.next();**

**numberOfPages = dataIn.nextInt();**

**books.addBook(new Book(ISBN, numberOfPages));**

**} catch (IllegalISBNException ex) {**

**System.out.println(**

**"Skipping book with ISBN = " + ISBN**

**+ " Invalid ISBN.\n" + ex);**

**}**

**}**

1. (5 pts) Note that the **Book** class implements the **Comparable<Book> i**nterface. However, the code for the **compareTo** method is missing. In the space below, write the missing code. Books should be compared on the basis of their ISBN. This means that if a list of books is sorted using the natural order as supplied by this **compareTo** method, the resulting list will be in ascending order by ISBN.

**@Override**

**public int compareTo(Book otherBook) {**

**return this.ISBN.compareTo(otherBook.ISBN);**

**}**

1. (5 pts) In the **BookList** class, provide the code for method **naturalSort**. This method sorts the books in the array list named **books** according to their natural order.

**public void naturalSort() {**

**Collections.sort(books);**

**}**

1. (10 pts) In the **BookList** class, provide the code for method **sortByPageCount**. This method sorts the books in the array list in ascending order by pageCount. You must use a **Comparator** for this method, and you must create an anonymous class. Do not create an external class.

**public void sortByPageCount() {**

**Collections.sort(books, new Comparator<Book>() {**

**@Override**

**public int compare(Book book01, Book book02) {**

**return book01.getPageCount() –**

**book02.getPageCount();**

**}**

**});**

**}**