

Write the following Java classes with the following behaviors. Zip your entire BlueJ project folder and save it as lastname.zip (use your own last name). Submit this zip file to BCIT's SHARE IN server (/in/comp/1451/assignment1 folder) before 11:59 PM, on the night before your class meets for its 6th week.

This assignment tests what we have learned in lessons 1, 2, 3, 4, and 5.

The purpose of this assignment is to create two Stores: a book store (class BookStore) and a shoe store (class ShoeStore).

Note the following:

Every instance variable must be very clearly named.

Every instance variable must have accessor and mutator methods.

Extensive JavaDoc comments must be provided as described in the lesson and Appendix J.

Constructors must have parameters for every instance variable, and must call its class's mutator methods to set the instance variables to the constructor's parameters.

Every subclass constructor must take parameters for its superclass's constructor, and call `super()` as its first instruction.

NOTE: some sample data is given in this assignment write-up, along with a sample run of how the program should run, given that sample data. However, your marker will actually test your assignment with other data...a larger sample of books and shoes.

I have put some code in here that you can copy/paste into your classes.

Contents

class BookStore	3
Sample BookStore Run.....	12
enum BookType	16
class ShoeStore	16
Sample ShoeStore Run.....	17
enum ShoeType	20
class Store	22
class Address.....	22
class Book.....	23
class Author.....	23
class Date	24
class Name	25
class Shoe.....	25
enum Material	25
class Item	26

class BookStore

This is a subclass of Store. It also has an instance variable for “specialty”, which is an enum of the following possible values: FICTION, NONFICTION, SCIENCEFICTION, REFERENCE

Note that we can map these enums to Strings (and vice versa) as follows in the BookType enum (see below):

```
import java.util.Map;
import java.util.HashMap;

public enum BookType {
    FICTION("fiction"), NONFICTION("nonfiction"), SCIENCEFICTION("sciencefiction"),
    REFERENCE("reference");

    private String theBookType;

    private static Map<String, BookType> lookup = new HashMap<String, BookType>();

    static{
        for (BookType b : BookType.values()){
            lookup.put(b.getTheBookType(), b);
        }
    }

    private BookType(String theBookType) {
        this.theBookType = theBookType;
    }

    public String getTheBookType() {
        return theBookType;
    }
}
```

```
}

// the following method allows me to create a BookType enum from a String!
// e.g. in the Book class, I could set the BookType instance variable to:
// bookType = BookType.get("fiction");
public static BookType get(String theBookType) {
    return lookup.get(theBookType);
}
}
```

Provide an overloaded constructor: one accepts Address, String, and BookType parameters for address, name, and specialty, respectively; the other accepts Address, String, and String parameters for address, name, and specialty, respectively – example: `this.specialty = BookType.get(specialty)`.

Both constructors then call an `addBooks()` method which creates anonymous objects and adds the following books to its (parent's) `HashMap`. The key is the Book Item's `uniqueID` String; and the value is the Book.

Kg	Manuf Price \$	Retail Price \$	ISBN	Author	Date Pub.	Title	Genre
0.4	2	4	1234	Jerome David Salinger (Jan 1, 1919) aka "JD"; fiction writer	May 14 1951	The Catcher in the Rye	fiction
1	11	12	2345	Jerome David Salinger (Jan 1, 1919) aka "JD"; fiction writer	Jan 31, 1948	A Perfect Day for Bananafish	fiction
2	33	35	3456	Jerome David Salinger (Jan 1, 1919) aka "JD"; fiction writer	Dec 12, 1945	A Boy in France	fiction
2.1	2	6	4567	Malcolm Gladwell (Sept 3, 1963); nonfiction writer	Nov 18, 2008	Outliers	nonfiction
0.5	3	5	5678	Malcolm Gladwell (Sept 3, 1963); nonfiction writer	Mar 1, 2000	The Tipping Point	nonfiction

0.01	4	4	6789	Frederik Pohl (Nov 26, 1919), aka "Paul Dennis Lavond"; science fiction writer	July 4, 1977	Gateway	science fiction
0.1	5	11	abcd	Frederik Pohl (Nov 26, 1919), aka "Paul Dennis Lavond"; science fiction writer	Oct 6, 1937	Elegy to a Dead Planet: Luna	science fiction
0.8	15	30	efgh	Richard Phillips Feynman (May 11, 1918), reference writer	May 20, 1942	Principle of Least Action in Quantum Mechanics	reference
0.6	44	45.5	ijkl	Richard Phillips Feynman (May 11, 1918), reference writer	June 30, 1964	The Messenger Lectures	reference

1.0	3	13	mnop	Richard Phillips Feynman (May 11, 1918), reference writer	Nov 1, 1985	Surely You're Joking Mr. Feynman	nonfiction
-----	---	----	------	---	-------------	----------------------------------	------------

The BookStore must provide the following methods (see the screenshots for example outputs). One of the methods has been filled in for an example.

```
import java.util.Collection;
import java.util.Iterator;
```

```
class BookStore extends Store
{
    private BookType specialty;
```

```
    public BookStore(Address address, String name, BookType specialty){
        super(address, name);
        this.specialty = specialty;
        addBooks();
    }
```

```
    public BookStore(Address address, String name, String specialty){
        super(address, name);
        this.specialty = BookType.get(specialty);
        addBooks();
    }
```

```
private void addBooks(){
    Date birthDate = new Date(1919, 1, 1);
    Name name = new Name("Jerome", "David", "Salinger");
    BookType genre = BookType.get("fiction");
    Author author = new Author(birthDate, name, genre, "JD");
    Date datePublished = new Date(1951, 5, 14);
    String title = "The Catcher in the Rye";
    Book b = new Book(0.4, 2.0, 4.0, "1234", author, datePublished, title, genre);
    addItem(b);

    datePublished = new Date(1948, 1, 31);
    title = "A Perfect Day for Bananafish";
    genre = BookType.get("fiction");
    b = new Book(1, 11, 12, "2345", author, datePublished, title, genre);
    addItem(b);

    datePublished = new Date(1945, 12, 12);
    title = "A Boy in France";
    genre = BookType.get("fiction");
    b = new Book(2, 33, 35, "3456", author, datePublished, title, genre);
    addItem(b);

    birthDate = new Date(1963, 9, 3);
    name = new Name("Malcolm", "Gladwell");
    genre = BookType.get("nonfiction");
    author = new Author(birthDate, name, genre);
    datePublished = new Date(2008, 11, 18);
    title = "Outliers";
    b = new Book(2.1, 2, 6, "4567", author, datePublished, title, genre);
    addItem(b);
}
```



```
datePublished = new Date(2000, 3, 1);  
title = "The Tipping Point";  
genre = BookType.get("nonfiction");  
b = new Book(0.5, 3, 5, "5678", author, datePublished, title, genre);  
addItem(b);
```

```
birthDate = new Date(1919, 11, 26);  
name = new Name("Frederik", "Pohl");  
genre = BookType.get("sciencefiction");  
author = new Author(birthDate, name, genre, "Paul Dennis Lavond") ;  
datePublished = new Date(1977, 7, 4);  
title = "Gateway";  
b = new Book(0.01, 4, 4, "6789", author, datePublished, title, genre);  
addItem(b);
```

```
datePublished = new Date(1937, 10, 6);  
title = "Elegy to a Dead Planet: Luna";  
genre = BookType.get("sciencefiction");  
b = new Book(0.1, 5, 11, "abcd", author, datePublished, title, genre);  
addItem(b) ;
```

```
birthDate = new Date(1918, 5, 11);  
name = new Name("Richard", "Phillips", "Feynman");  
genre = BookType.get("reference");  
author = new Author(birthDate, name, genre) ;  
datePublished = new Date(1942, 5, 20);  
title = "Principle of Least Action in Quantum Mechanics";  
b = new Book(0.8, 15, 30, "efgh", author, datePublished, title, genre);  
addItem(b);
```

```
datePublished = new Date(1964, 6, 30);
title = "The Messenger Lectures";
genre = BookType.get("reference");
b = new Book(0.6, 44, 45.5, "ijkl", author, datePublished, title, genre);
addItem(b);
```

```
datePublished = new Date(1985, 11, 1);
title = "Surely You're Joking Mr. Feynman";
genre = BookType.get("nonfiction");
b = new Book(1.0, 3, 13, "mnop", author, datePublished, title, genre);
addItem(b);
}
```

```
public void displayAllBooksWrittenByAuthorsOverThisAge(int ageInYears){
    Collection<Book> books      = getCollectionOfItems(); // From the Store class
    Iterator<Book> it           = books.iterator();
    boolean displayedSome      = false;
    while(it.hasNext()){
        Book b = it.next();
        int ageYears = b.getDatePublished().getYear() - b.getAuthor().getBirthDate().getYear();
        if(ageYears > ageInYears){
            System.out.println(b.getTitle() + " was written by " +
                b.getAuthor().getName().getLastName() + " at age " + ageYears +
                ", which is more than " + ageInYears);
            displayedSome = true;
        }
    }
    if(displayedSome == false){
        System.out.println("No books by authors over age " + ageInYears);
    }
}
...all other methods go here too...
```

}

Sample BookStore Run

Test a BookStore named “Chapters”, with an address of “1234 Main Street, Vancouver, A3A A4A”, and a specialty of “fiction”.

displayAllBooksByEveryAuthor()

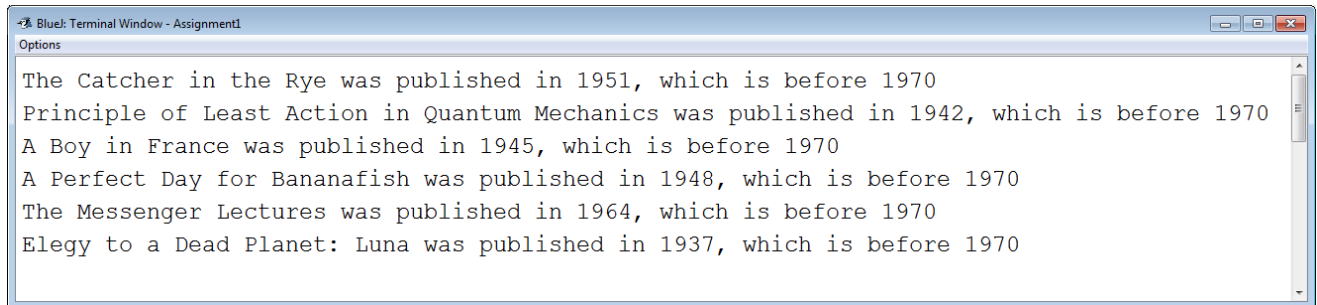
A terminal window titled "BlueJ: Terminal Window - Assignment1" with an "Options" menu. The output of the `displayAllBooksByEveryAuthor()` method is displayed as a list of author and book information:

```
Gladwell wrote The Tipping Point in 2000
Salinger wrote The Catcher in the Rye in 1951
Feynman wrote Principle of Least Action in Quantum Mechanics in 1942
Pohl wrote Gateway in 1977
Salinger wrote A Boy in France in 1945
Salinger wrote A Perfect Day for Bananafish in 1948
Feynman wrote The Messenger Lectures in 1964
Feynman wrote Surely You're Joking Mr. Feynman in 1985
Pohl wrote Elegy to a Dead Planet: Luna in 1937
Gladwell wrote Outliers in 2008
```

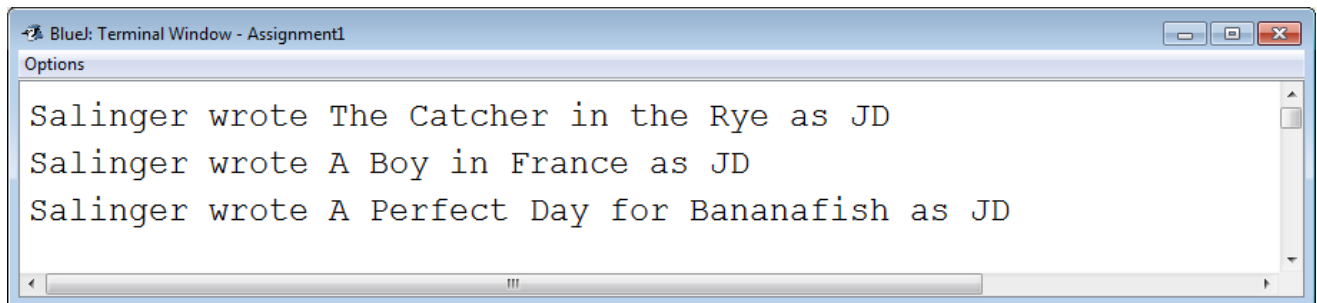
displayAllBooksByAuthor(String lastname)

A terminal window titled "BlueJ: Terminal Window - Assignment1" with an "Options" menu. The output of the `displayAllBooksByAuthor(String lastname)` method, where the lastname is "Salinger", is displayed as a list of books:

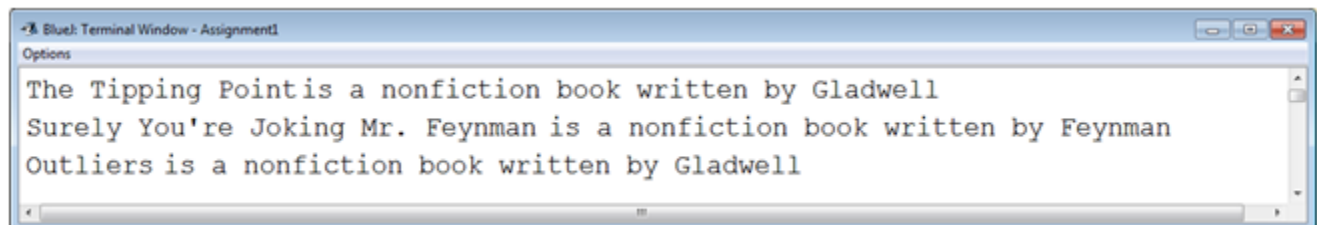
```
Salinger wrote The Catcher in the Rye
Salinger wrote A Boy in France
Salinger wrote A Perfect Day for Bananafish
```

displayAllBooksWrittenBefore(int year)

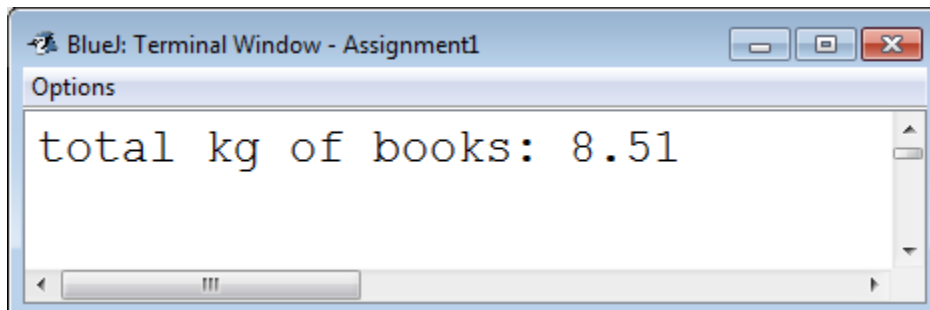
```
BlueJ: Terminal Window - Assignment1
Options
The Catcher in the Rye was published in 1951, which is before 1970
Principle of Least Action in Quantum Mechanics was published in 1942, which is before 1970
A Boy in France was published in 1945, which is before 1970
A Perfect Day for Bananafish was published in 1948, which is before 1970
The Messenger Lectures was published in 1964, which is before 1970
Elegy to a Dead Planet: Luna was published in 1937, which is before 1970
```

displayTitlesOfBooksWrittenBy(String pseudonym)

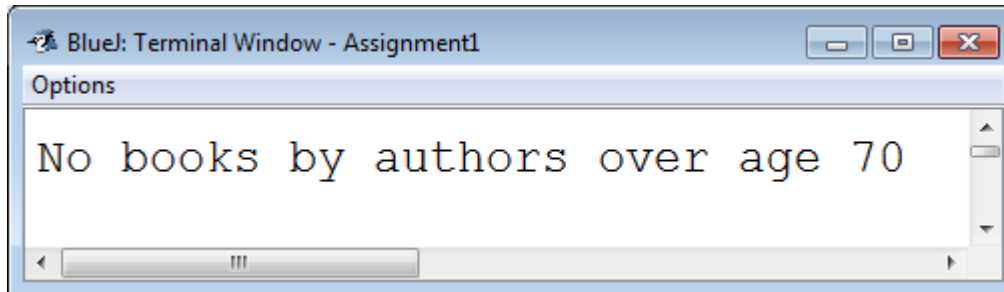
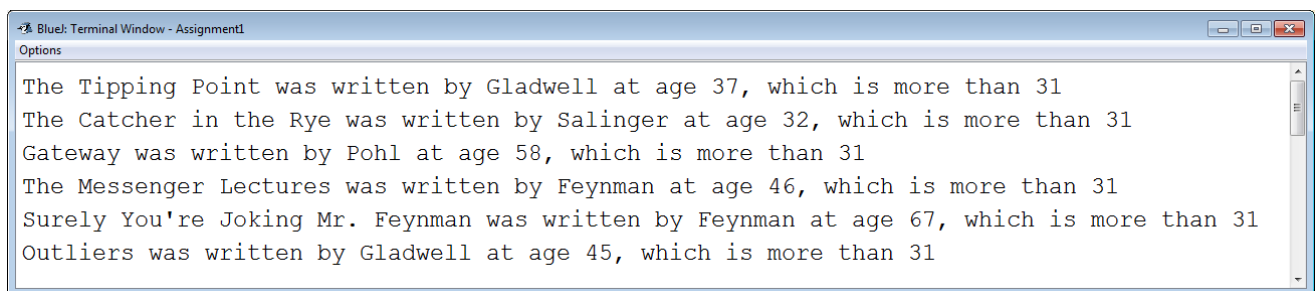
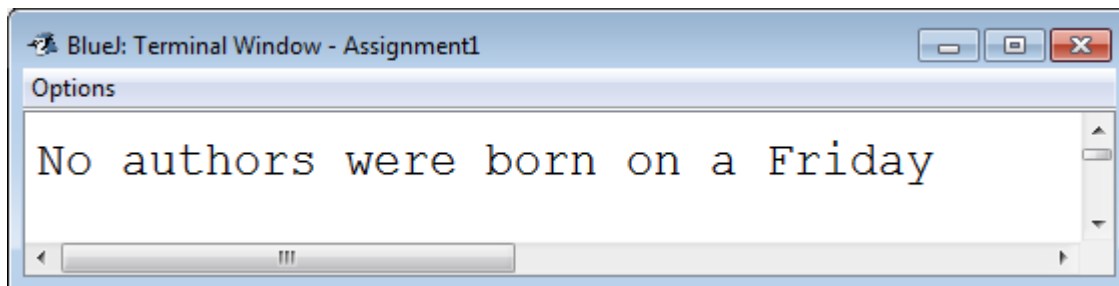
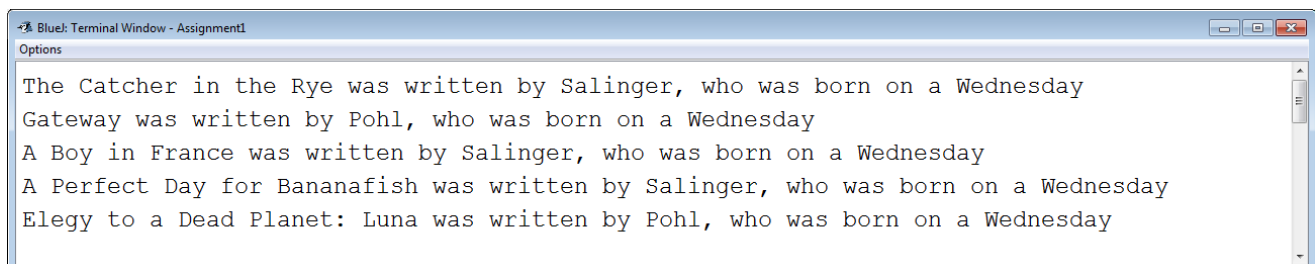
```
BlueJ: Terminal Window - Assignment1
Options
Salinger wrote The Catcher in the Rye as JD
Salinger wrote A Boy in France as JD
Salinger wrote A Perfect Day for Bananafish as JD
```

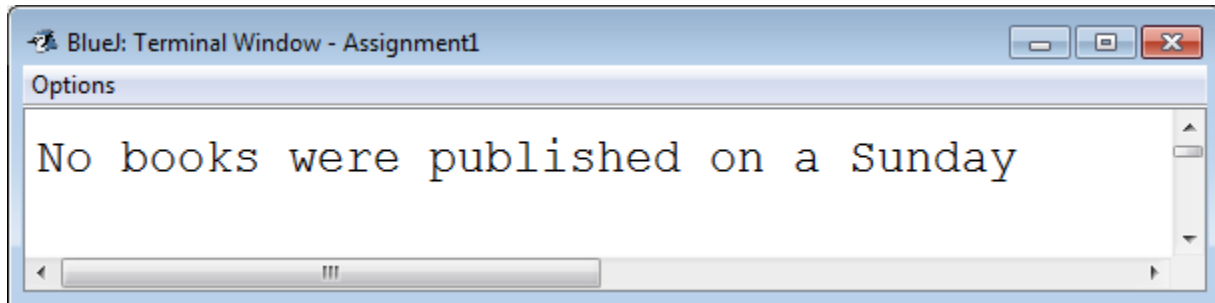
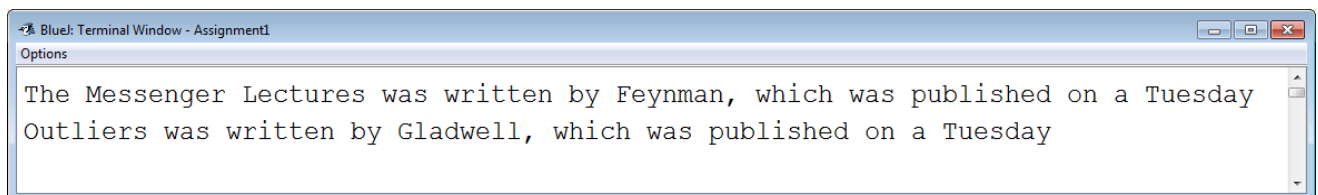
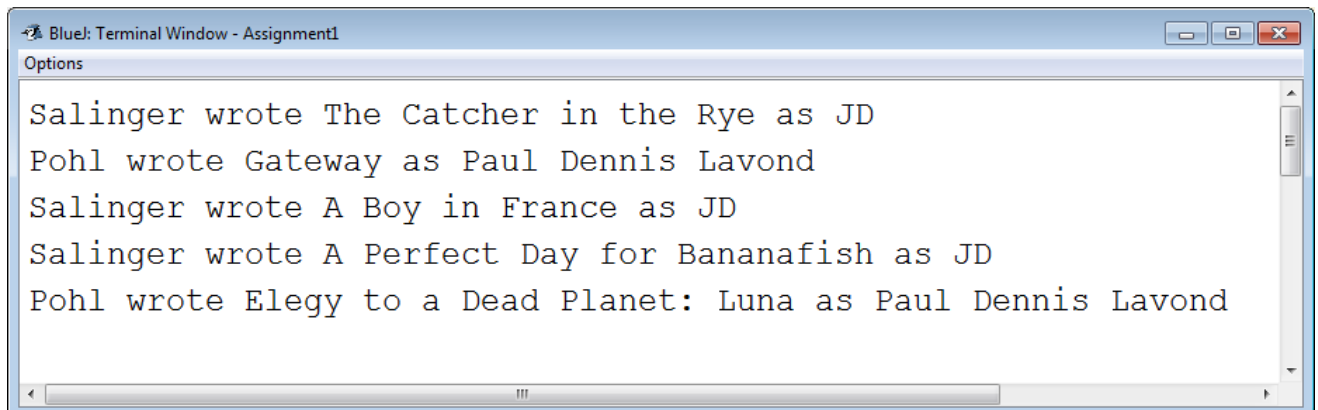
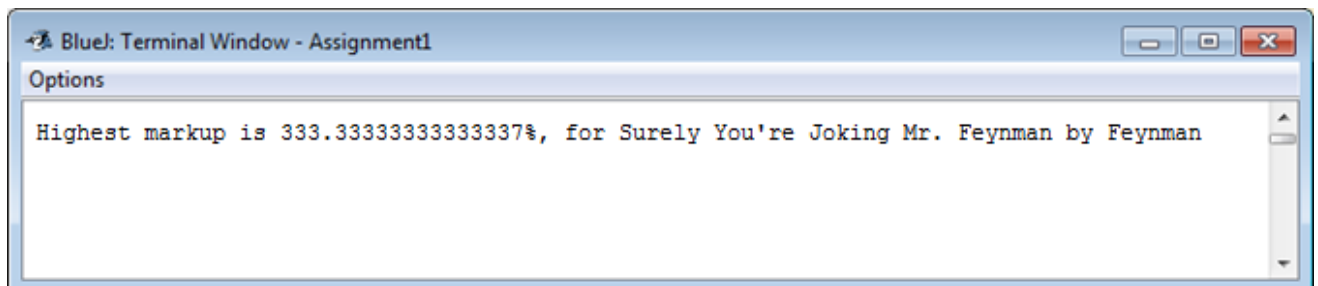
displayAllBooksForGenre(String genre)

```
BlueJ: Terminal Window - Assignment1
Options
The Tipping Point is a nonfiction book written by Gladwell
Surely You're Joking Mr. Feynman is a nonfiction book written by Feynman
Outliers is a nonfiction book written by Gladwell
```

displayTotalWeightKgOfAllBooks()

```
BlueJ: Terminal Window - Assignment1
Options
total kg of books: 8.51
```

displayAllBooksWrittenByAuthorsOverThisAge(int ageInYears)**...Or...****displayAllBooksWrittenByAuthorsBornOn(String dayOfTheWeek)****...Or...**

displayAllBooksPublishedOn(String dayOfTheWeek)**...Or...****displayAllBooksWrittenByAuthorsWithAPseudonym()****displayBookWithBiggestPercentageMarkup()****displayAllBooksWrittenOutsideSpecialty()**

```
Blue: Terminal Window - Assignment1
Options
Feynman usually wrote reference but wrote Surely You're Joking Mr. Feynman which is nonfiction
```

enum BookType

see above

class ShoeStore

This is a subclass of Store. It also has an instance variable for “department”, which is an enum of the following values: WOMEN, MEN, CHILDREN, SPORTS, DRESS (created similarly to BookType, above).

The ShoeStore class must have similar functionality as the BookStore (above). Its instance variables are:

Department (see above).

The inventory that will be created in the addShoes() method is as follows:

Kg	Manuf Price \$	Retail Price \$	ID	Designer	Size	Material	Color	Department
1	58.5	90	Diameter	Skechers	10	LEATHER	DARK_GRAY	men
1.15	104	160	Wave	Robert Cobbler	12	LEATHER	BLACK	dress
1	110.5	170	Monet	Geox	7	CLOTH	BLUE	men
0.85	84.5	130	Camya Multi Glitter	Nine West	8	PLASTIC	BLACK	women
0.9	97.5	150	Marieclaire	Geox	10	PLASTIC	GRAY	women
0.6	45.5	70	Balance Of The Force	Stride Rite	9	RUBBER	GRAY	children
0.7	39	60	Top-Sider Unisex Bluefish H&L	Sperry	9	CLOTH	ORANGE	children
0.85	32.5	50	Lite Kicks Rainbow Sprite	Skechers	10	PLASTIC	PINK	children
0.5	39	60	Toachi	Robert Cobbler	5	CLOTH	BLUE	children
1.2	117	180	Jordan Ace 23 II	Nike	13	RUBBER	WHITE	sports

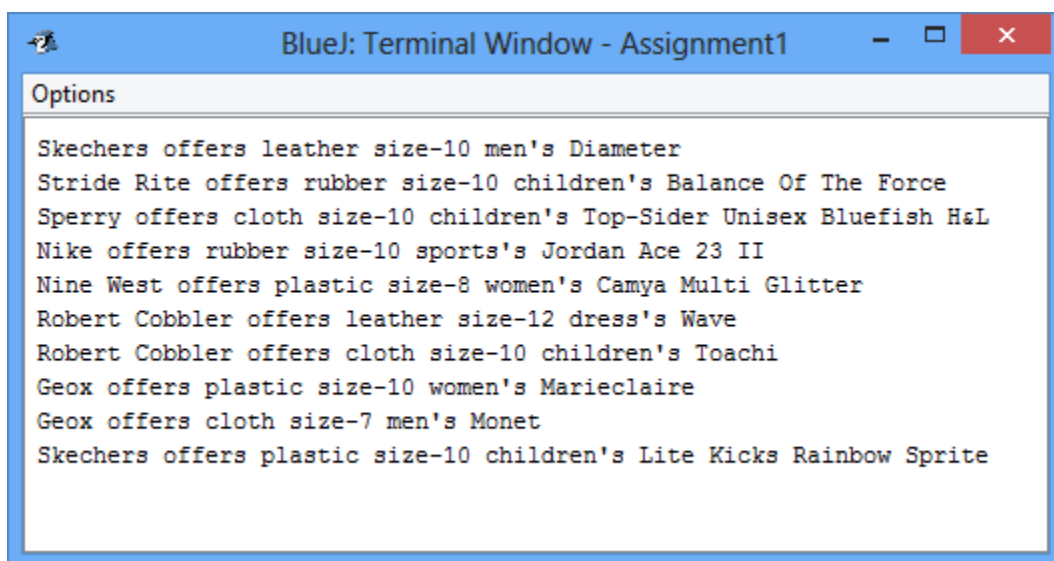
This class's methods must have the same sort of output as the BookStore methods.

```
displayAllShoesAndDesigners()
displayAllShoesByDesigner(String designerName)
displayAllShoesMadeOf(Material material)
displayAllShoesMadeOf(String material)
displayNumberOfShoesDesignedBy(Name designer)
displayNumberOfShoesDesignedBy(String designerLastName)
displaySmallestShoeSize ()
displayTotalWeightKgOfAllShoes()
displayAllShoesOfThisMaterialMadeByThisDesigner(Material m, Name designer)
displayAllShoesNotInMatchingStore() // e.g. For a shoe store with department
"WOMEN", show all the shoes of type MEN, CHILDREN, SPORTS, and DRESS
```

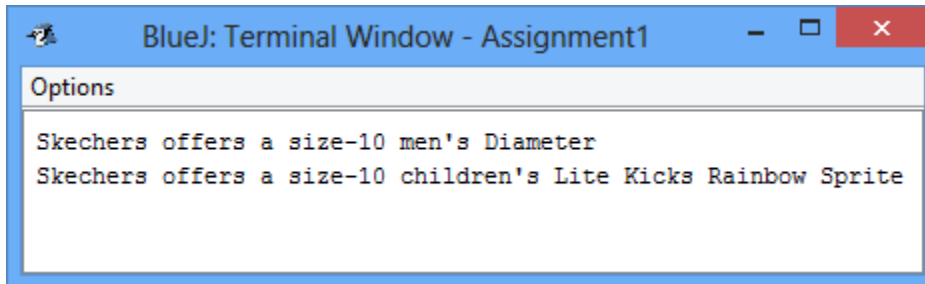
Sample ShoeStore Run

Test a ShoeStore named "My Shoes", with an address of "789 East 1st Street, West Vancouver, V3A 7A4", which specializes in selling children's shoes.

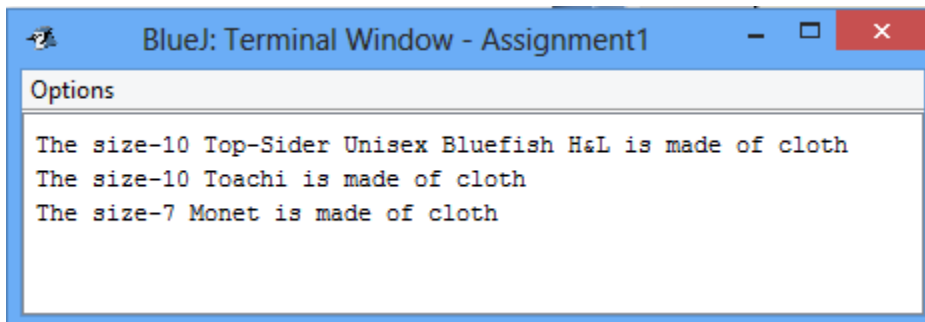
displayAllShoesAndDesigners()



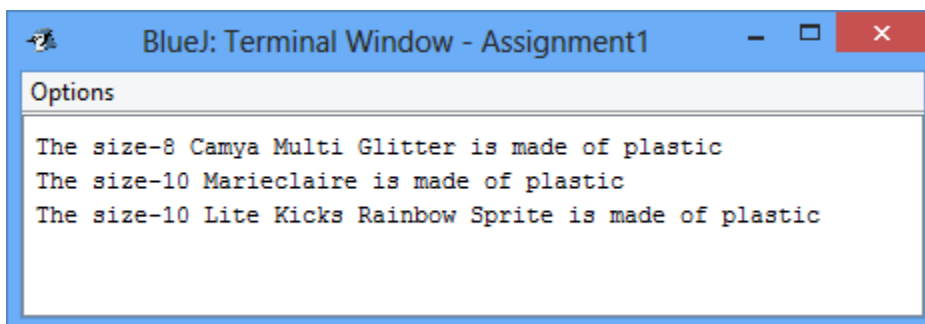
```
Options
Skechers offers leather size-10 men's Diameter
Stride Rite offers rubber size-10 children's Balance Of The Force
Sperry offers cloth size-10 children's Top-Sider Unisex Bluefish H&L
Nike offers rubber size-10 sports's Jordan Ace 23 II
Nine West offers plastic size-8 women's Camya Multi Glitter
Robert Cobbler offers leather size-12 dress's Wave
Robert Cobbler offers cloth size-10 children's Toachi
Geox offers plastic size-10 women's Marieclaire
Geox offers cloth size-7 men's Monet
Skechers offers plastic size-10 children's Lite Kicks Rainbow Sprite
```

displayAllShoesByDesigner(String designerName)

```
Options
Skechers offers a size-10 men's Diameter
Skechers offers a size-10 children's Lite Kicks Rainbow Sprite
```

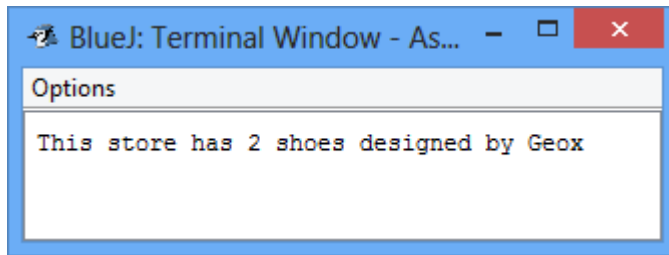
displayAllShoesMadeOf(Material material)

```
Options
The size-10 Top-Sider Unisex Bluefish H&L is made of cloth
The size-10 Toachi is made of cloth
The size-7 Monet is made of cloth
```

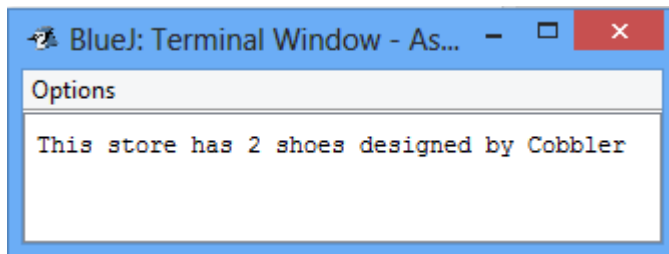
displayAllShoesMadeOf(String material)

```
Options
The size-8 Camya Multi Glitter is made of plastic
The size-10 Marieclaire is made of plastic
The size-10 Lite Kicks Rainbow Sprite is made of plastic
```

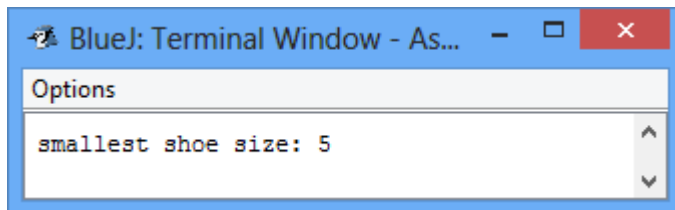
displayNumberOfShoesDesignedBy(Name designer)



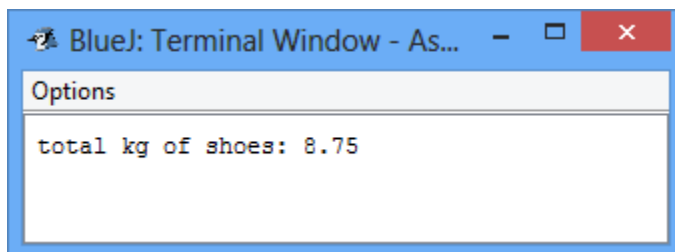
displayNumberOfShoesDesignedBy(String designerLastName)



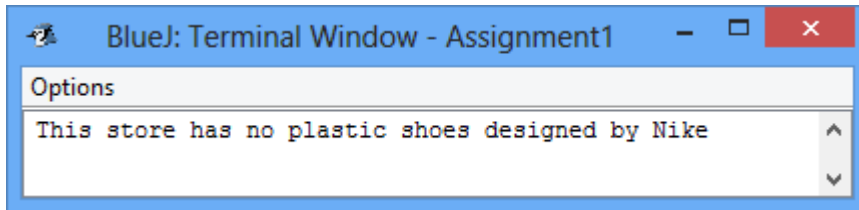
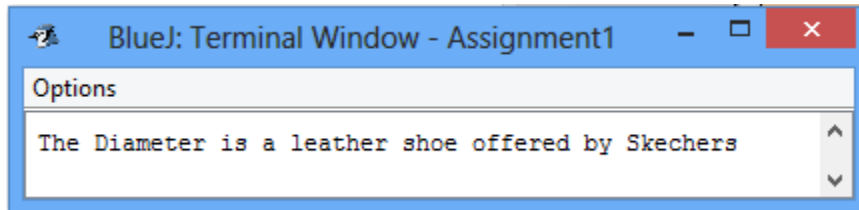
displaySmallestShoeSize ()



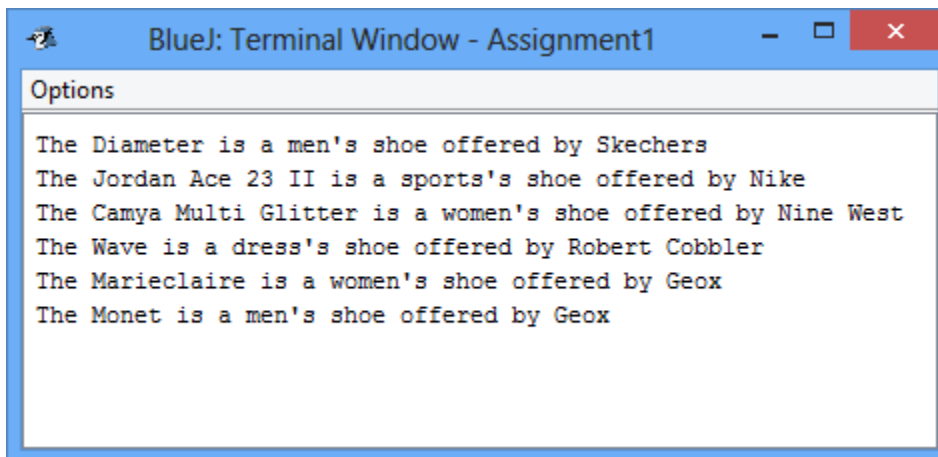
displayTotalWeightKgOfAllShoes()



displayAllShoesOfThisMaterialMadeByThisDesigner(Material m, Name designer)



displayAllShoesNotInMatchingStore()



If it is still unclear what **displayAllShoesNotInMatchingStore()** does, please complete BookStore **displayAllBooksWrittenOutsideSpecialty()** first. The two methods behave the same. [Shoe.type](#) is to [ShoeStore.department](#) as [Book.genre](#) is to [Book.author.specialty](#).

enum ShoeType

see ShoeStore, above, for the enums

class Store

This is the parent class of all other Stores.

It has the following instance variables:

A `streetAddress`, of class `Address`.

A `name`, of class `String`.

The store contains a `HashMap` of `Items`. The key of each item is its `uniqueID` `String`. For example, an `Item` may have a `uniqueID` of “abc123”. The value is the item itself.

This class offers very useful, generic methods such as:

```
public void addItem(Item item){
    itemsForSale.put(item.getUniqueID(), item);
}

public Item getItemByKey(String key){
    return itemsForSale.get(key);
}

public Collection getCollectionOfItems(){
    return itemsForSale.values();
}
```

class Address

Contains instance variables for `street number` (`String`), `street name` (`String`), `city` (`String`), `postal code` (`String`).

class Book

Has instance variables for author (class Author), datePublished (class Date), title (String), genre (bookType enum). Note that a Book also has an ISBN (a String). In our class we will NOT have an ISBN instance variable. Our Book class in fact uses its parent class's uniqueID variable and methods: e.g. class Book has methods called setISBN() and getISBN(), but in turn it just calls setUniqueID() and getUniqueID() from the Item class. This class also has a getYearPublished() convenience function, as well as getAuthorFullName(), and getGenreString() which returns a String such as "fiction" when its BookType is BookType.FICTION (return genre.getTheBookType();). Book is a subclass of class Item.

class Author

Has instance variables for birthDate (class Date), name (class Name), genre (enum BookType), and pseudonym (String).

class Date

Has instance variables for year (int), month (int), dayOfTheMonth (int). Has accessor methods for each, plus getMonthName() (e.g. returns "January"), getSuffix(e.g. returns "st" for 1, "nd" for 2, "rd" for 3, "th" for 4, etc...), and getDayOfWeek() (e.g. returns "Friday"). Here is an algorithm to get the day of the week from a date:

Example date: **May 17th, 1989**

- Step 1: determine how many twelves fit in the last two digits of the year
e.g. there are **7** twelves in 89 ($12 * 7 = 84$) **Sum is 7**
- Step 2: determine the remainder from step 1
e.g. $89 - 84 = 5$ remainder **Sum is 7 + 5**
- Step 3: determine how many fours fit into the remainder from step 2
e.g. **1** four fits into five **Sum is 7 + 5 + 1**
- Step 4: Add the date of the month
e.g. **17** for the 17th **Sum is 7 + 5 + 1 + 17**
- Step 5: Add the month code (see below)
e.g. **2** for May **Sum is 7 + 5 + 1 + 17 + 2**
- Step 6: Mod the sum by 7
 $7 + 5 + 1 + 17 + 2 = 32$; $32 \% 7 = 4 = \text{WEDNESDAY}$

Match this number with the day of the week:

Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday
0	1	2	3	4	5	6

Month codes:

January: 1	February: 4	March: 4
April: 0	May: 2	June: 5
July: 0	August: 3	September: 6
October: 1	November: 4	December: 6

For dates in the 1700s, add 4. For dates in the 1800s, add 2. For dates in the 1900s, add nothing. For dates in the 2000s, add 6. For any other years, simply return “Dunno”.

Add 6 for January or February of a leap year.

The mutators do not permit invalid dates such as February 30th.

class Name

Has instance variables for first name (String), middle name (String), last name (String). Overload the constructor to allow for people that have one (e.g. “Madonna” or “Cher”), or two (e.g. “Tiger Woods”), or three names (e.g. “Andrew Lloyd Webber”).

In addition to all regular accessor and mutator methods, include a method `getFullName()` which returns the full name (such as “Madonna”, or “Tiger Woods”, or “Andrew Lloyd Webber”).

class Shoe

Has instance variables for material (enum Material), size (int), designer (Name class; e.g. “Nike” or “Manolo Blahnik”), shoe type (enum), and color (class `java.awt.Color`; for example, `java.awt.Color.gray`). Shoe is a subclass of class Item. Note that a Shoe also has a description (a String). In our class we will NOT have an description instance variable. Our Shoe class in fact uses its parent class’s `uniqueID` variable and methods: e.g. class Shoe has methods called `setDescription()` and `getDescription()`, but in turn it just calls `setUniqueID()` and `getUniqueID()` from the Item class.

enum Material

Has values for PLASTIC, LEATHER, RUBBER, CLOTH

class Item

Has instance variables for weightKg (double), manufacturingPriceDollars (double), suggestedRetailPriceDollars (double), uniqueID (String).

Start early. There are lots of classes to write:

