Create your First class

Estimated time needed: 30 minutes

In this lab, you will learn how to create a class and its instances in Java.

You are currently viewing this lab in a cloud-based Integrated Development Environment (Cloud IDE). This fully online IDE is pre-installed with JDK 21 and allows you to code, develop, and learn in one location.

Learning Objectives

After completing this lab, you will be able to:

- Create a class and define its attributes and methods
- Learn how to provide different levels of access to the members of a class
- Understand how to create instances of the class
- Compare the attributes of different instances of the same class
- Accept user input for the attributes of an instance

Creating a class

In Java, `Class` is like a template upon which you can create objects. It is a blueprint that broadly defines an object's characteristics and what you can do with it. In this lab, you will create a `Book` class, define its attributes and methods, and then create instances or `Objects` of `Book` class.

1. Create a project directory by running the following command.

   ```
   mkdir my_class_proj
   ```

2. Run the following code to create the directory structure.

   ```
   mkdir -p my_class_proj/src
   mkdir -p my_class_proj/classes
   mkdir -p my_class_proj/test
   cd my_class_proj
   ```

3. Now create a file named `Book.java` inside the src directory.

```
touch /home/project/my_class_proj/src/Book.java
```

4. Click on the button below to open the file for editing.

Open **Book.java** in IDE

5. Read each statement in the following program and understand how the attributes and functions of `Book` class are defined.

Paste the following content in `Book.java`.

```
public class Book {
    private String title;
    private String author;
    private float price;
    public void setTitle(String title) {
        this.title = title;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public void setPrice(float price) {
        this.price = price;
    }
    public String getTitle() {
        return this.title;
    }
    public String getAuthor() {
        return this.author;
    }
    public float getPrice() {
        return this.price;
    }
    public String toString() {
        return "Title - " + this.title + "\nAuthor - "
            + this.author + "\nPrice - "+ String.format("%.2f", this.price);
    }
}
```

In this code you are defining a class named `Book` with three attributes - title, author, and price.
Pay attention to the words before the variable name that defines attribute. These define:

- the access of the attribute
- the type of the attribute (can be built-in such as int, String, and so on, or user-defined)
  `private String title;` - `title` is a String attribute with private access. It can only be accessed from within the class.
  `private String author;` - `author` is a String attribute with private access. It can only be accessed from within the class.
  `private float price;` - `price` is a float attribute with private access. It can only be accessed from within the class.

`public void setTitle(String title)` - `setTitle` is a public setter method for the `title` attribute.

`public void setAuthor(String author)` - `setAuthor` is a public setter method for the `author` attribute.

`public void setPrice(float price)` - `setPrice` is a public setter method for the `price` attribute.

`public String getTitle()` - This is a public getter method for getting the title. You can also see the other attributes have getter methods. These help in retrieving the values of the attributes.

`public String toString()` - This method is an override of the toString() method from the Object class in Java. It is used to provide a custom string representation of an object. It returns a string concatenating the attributes. String.format("%.2f", this.price) ensures that the price is displayed with two decimal places.

6. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

   ```
   javac -d classes src/Book.java
   ```

7. Set the `CLASSPATH` variable.

   ```
   export CLASSPATH=$CLASSPATH:/home/project/my_class_proj/classes
   ```

8. Now you cannot run the Java program, as it does not have `main` method. `main` method is the entry point to any runnable Java program. It is a good programming practice to separate the classes from the runnable Java program. Create a new file named `BookAccess.java`.

   ```
   touch /home/project/my_class_proj/src/BookAccess.java
   ```

9. Click on the button below to open the file for editing.

Open **BookAccess.java** in IDE

10. Read each statement in the following program and understand how the instances or objects of `Book` class is created.

Paste the following content in `BookAccess.java`.

```
public class BookAccess {
    public static void main(String s[]) {
        Book book1 = new Book();
        book1.setTitle("Atomic Habits");
        book1.setAuthor("James Clear");
        book1.setPrice(30.00f);
        Book book2 = new Book();
        book2.setTitle("Sapiens");
        book2.setAuthor("Yuval Noah Harari");
        book2.setPrice(25.00f);
        System.out.println("The first book object is ");
        System.out.println(book1);
        System.out.println("The second book object is ");
        System.out.println(book2);
    }
}
```

11. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/BookAccess.java
```

```
java BookAccess
```

You will see the output as below:

```
The first book object is
Title - Atomic Habits
Author - James Clear
Price - 30.00
The second book object is
Title - Sapiens
Author - Yuval Noah Harari
Price - 25.00
```

Add method for price comparison

You will now use `Relational operators` that return a boolean value to compare the price of two books.

1. Click the button below to open `BookAccess.java`for editing.

Open **BookAccess.java** in IDE

2. Add the following code within the `BookAccess` class in place of the existing code.

```
public class BookAccess {
    private static void comparePrices(Book book1, Book book2) {
        float book1Price = book1.getPrice();
        float book2Price = book2.getPrice();
        String priceCmp = "";
        if (book1Price < book2Price) {
            priceCmp = book1.getTitle()+" costs less than " + book2.getTitle();
        } else if (book1Price == book2Price) {
            priceCmp = book1.getTitle()+" costs as much as " + book2.getTitle();
        } else {
            priceCmp = book1.getTitle()+" costs more than " + book2.getTitle();
        }
        System.out.println(priceCmp);
    }
    public static void main(String s[]) {
        Book book1 = new Book();
        book1.setTitle("Atomic Habits");
        book1.setAuthor("James Clear");
        book1.setPrice(30.00f);
        Book book2 = new Book();
        book2.setTitle("Sapiens");
        book2.setAuthor("Yuval Noah Harari");
        book2.setPrice(25.00f);
        System.out.println("The first book object is ");
        System.out.println(book1);
        System.out.println("The second book object is ");
        System.out.println(book2);
        comparePrices(book1, book2);
    }
}
```

`private static void comparePrices(Book book1, Book book2)` - `comparePrices` is a private static method which takes book1 and book2 as parameters. It compares the prices and prints a statement about the prices of the books.

3. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/BookAccess.java
```

4. Now run the Java program.

```
java BookAccess
```

You will see the output as below:

```
The first book object is
Title - Atomic Habits
Author - James Clear
Price - 30.00
The second book object is
Title - Sapiens
Author - Yuval Noah Harari
Price - 25.00
Atomic Habits costs more than Sapiens
```

Provide user menu

You will now create a command line app with array, loops and `Scanner`, a class from the util package that allows you to take user input. This code will implement a simple console-based menu system to manage books using an array of Book objects. The user can view or add books, and exit the program when desired.

1. Create a file named `BooksMenu.java` inside the src directory.

   ```
   touch /home/project/my_class_proj/src/BooksMenu.java
   ```

2. Click on the button below to open the file for editing.

Open **BooksMenu.java** in IDE

3. Read each statement in the following program and understand how the Books array is created, books are added to the array and books are retrieved. Paste the following content in `BooksMenu.java`.

   ```
   import java.util.Scanner;
   public class BooksMenu {
       public static void main(String s[]) {
           Scanner scanner = new Scanner(System.in);  // Create a Scanner object
           Book[] books = new Book[10];
           int bkIdx = 0;
           while(true) {
               System.out.println("Press 1 to view books, 2 to add books, any other key to exit");
               String userAction = scanner.nextLine();
               if (userAction.equals("1")) {
                   for(int i=0;i<books.length;i++) {
                       if(books[i] != null) {
                           System.out.println(books[i]);
                       }
                   }
               } else if (userAction.equals("2")) {
                   if(bkIdx == 10) {
                       System.out.println("10 books added already. Cannot add any more books!");
                       continue;
                   }
                   System.out.println("Enter book title");
                   String tmpTitle = scanner.nextLine();
                   System.out.println("Enter book author");
                   String tmpAuthor = scanner.nextLine();
                   System.out.println("Enter book price");
                   float tmpPrice = Float.parseFloat(scanner.nextLine());
                   Book bkTmp = new Book();
                   bkTmp.setTitle(tmpTitle);
                   bkTmp.setAuthor(tmpAuthor);
                   bkTmp.setPrice(tmpPrice);
                   books[bkIdx++] = bkTmp;
               } else {
                   break;
   ```

```
                }
            }
        }
    }
```

`import java.util.Scanner;` - Allows the program to use the Scanner class for taking input from the user.
`Scanner scanner` - Used to read user input from the console.
`Book[] books = new Book[10]` - Creates an array of size 10 to store up to 10 Book objects. At a later point when you learn about collections, you can replace the fixed size array with an expandable collection.
`int bkIdx = 0` - Keeps track of how many books have been added to the books array.

`while(true)` - The program runs an infinite loop to continuously display the menu until the user chooses to exit. The user is prompted to press:

- `1` To view books.
- `2` To add books.
- Any other key to exit.

If the user presses 1, the program iterates over the books array.
If a Book object is not null, its details are printed.

If the user presses 2:
The program checks if bkIdx has reached 10 (array is full). If so, it prevents further additions. Else it prompts the user for the book's:
Title (tmpTitle)
Author (tmpAuthor)
Price (tmpPrice, converted from string to float using Float.parseFloat).
Creates a new Book object and stores it in the books array at the bkIdx position.
Increments bkIdx to prepare for the next addition.

4. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

   ```
   javac -d classes src/BooksMenu.java
   ```

5. Now run the Java program.

   ```
   java BooksMenu
   ```

As the output for this is dependent on the user input, a sample of the same has been given below.

```
Press 1 to view books, 2 to add books, any other key to exit
2
Enter book title
The Power of Nothingness
Enter book author
Alexandra David-Neel
Enter book price
32.00
Press 1 to view books, 2 to add books, any other key to exit
1
Title - The Power of Nothingness
Author - Alexandra David-Neel
Price - 32.00
Press 1 to view books, 2 to add books, any other key to exit
```

Practice Exercise

1. Add a method to BooksMenu.java to compare the price of two books and return the most expensive of the two.

2. Add a menu item 3, to compare the price of the books which will take the index of the books in the array and compare the prices.

   Hint: Use Scanner and Integer.parseInt to get the index number

▶ Click here for solution
▶ Click here for sample output

Conclusion

In this lab, you learned how to create class and define its attributes and add appropriate methods and to create one or more objects of the class.

Author(s)

Lavanya