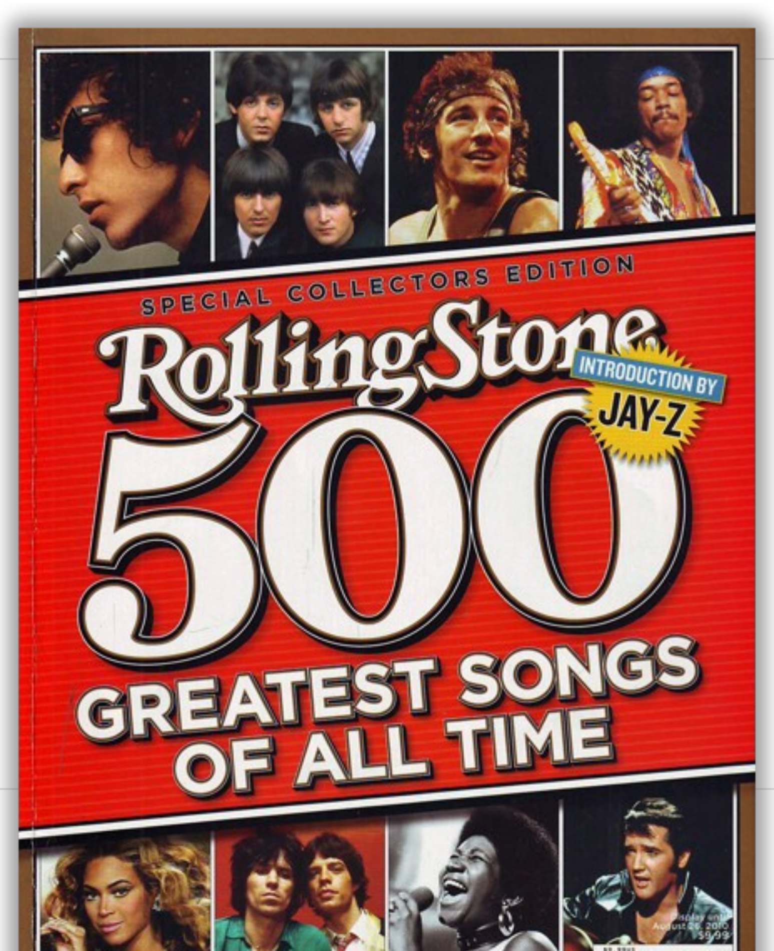
ct count down.

DO
NE

PROGRAM DESCRIPTION



Write a program that:

1. Reads a file containing a list of Rolling Stone magazine's top songs of all time.
2. Prints them out as a countdown list (descending order).

The program has a number of purposes. You will:

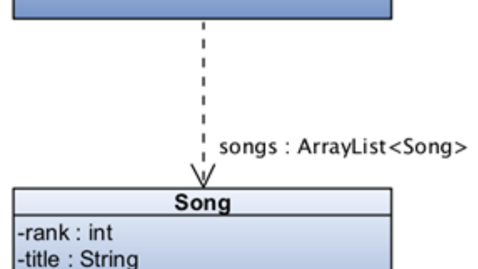
1. Learn how to read data from a file.
2. Learn how to split strings into parts.
3. Learn how to create lists of objects.
4. Learn how to build and output results.
5. Learn how to run unit tests to verify and validate your program.

TASKS & REQUIREMENTS

1. Read the **workspace-cs106-v3 README.md** so that you know about what the workspace includes and how to use the `sbcc.Core` functions (they make Java programming **much** easier).
2. In Eclipse's Package Explorer:
    1. Right-click on the `JavaCoreTemplate` project and select Copy.

       NOTE: if you don't see `JavaCoreTemplate` in the Package Explorer, then you likely didn't **setup the workspace-cs106-v3**, or you might have used Windows' built-in zip extractor, which, by default, extracts the workspace folder into a higher-level workspace folder with the same name. The setup link shows the proper steps.
    2. Right-click in an empty area of the Package Explorer and select Paste.

       The new project's name must follow this pattern: **{FLname}_RockCountdown_{term}**, where {FLname} is replaced by the first letter of your first name plus your last name, and {term}. E.g. if it's Fall 2025 and your name is Maria Marciano, your project name must be MMarciano_RockCountdown_F25. If your project name does not follow this pattern, it will not be graded.
3. Change the package called "`changemyname`" to `rockcountdown`.
4. You will be implementing the code for the following class diagram:



5. Create the `Song` class:
    1. Use the New Class wizard to create a new class.
        1. Set the Package to `rockcountdown`.
        2. Set the Name to `Song`.
        3. Uncheck "public static void main...".
    2. Add the rank, title, and artist fields to the `Song`.
    3. Add getters and setters to the properties to make them properties (note that a property is a field plus a getter and/or setter). **Productivity tip**: Right-click in the source code editor | Source... | Generate getters and setters...
    4. Add a constructor to initialize the fields. **Tip**: Right-click in the source code editor | Source... | Generate Constructor using Fields...
6. ✓ Ensure that `Song` objects can be created an initialized:
    1. ✓ Create a new package under the src folder called `sbccunittest`.
    2. Download (right-click Save Link As...) **SongListTester.java** into `sbccunittest`. Right-click on your project, then choose Refresh to show the new file as in your project.
        1. `SongListTester` is a JUnit test class.
        2. Its job is to test various aspects of your program.
        3. The JUnit view will tell you which tests passed and failed.
        4. ✓ `SongListTester` will output your program's score in the console view.
    3. ✓ Run | Run As | JUnit Test.
    4. ✓ If you've coded Song up correctly, the `testNewSongFromFields`() test will pass, and the JUnit view in Eclipse will show a green bar. The console will show your score. If there are any problems, fix them so that `testNewSongFromFields` passes.
7. Allow a new `Song` object to be created from a tab delimited string.

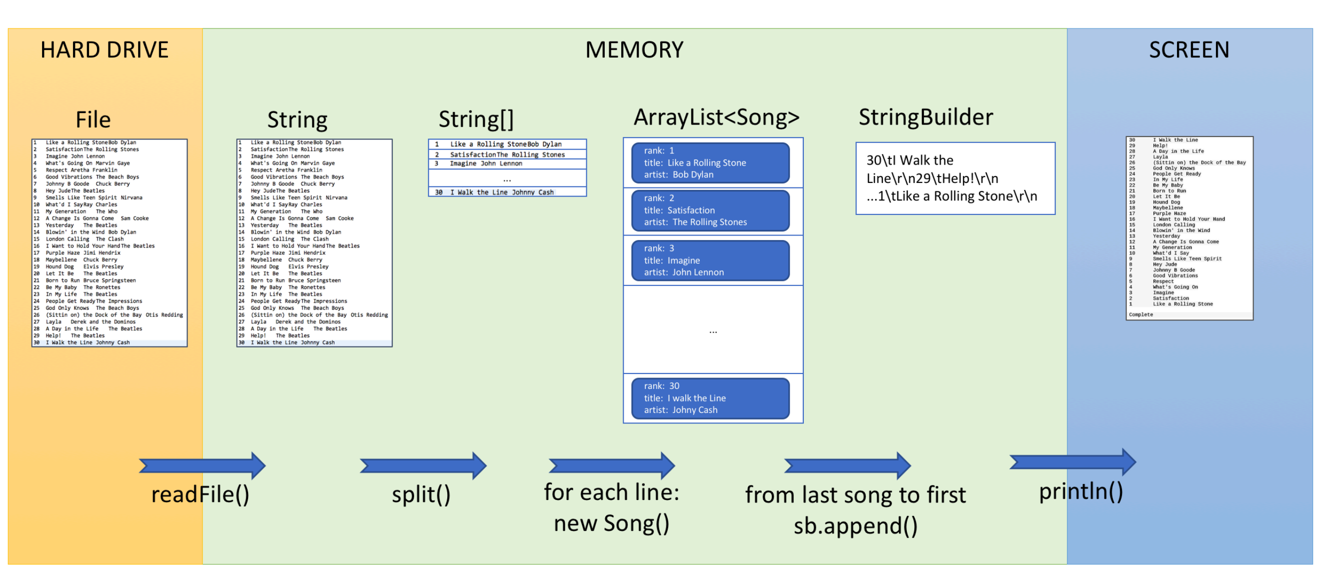
   ok. mann.
    1. Add the one-string constructor to `Song`:
        1. The idea of the one-string constructor is that the caller passes in a tab delimited string containing the `Song` properties. Here is an example of the string:

           `"21	Born to Run	Bruce Springsteen"`

           \t \t

           The constructor splits the `String` up into tokens and sets the fields equal to the corresponding token.
        2. Right-click in the editor | Source | Generate constructor from superclass...
        3. ✓ OK.
        4. ✓ Note that this creates a constructor for you that takes no parameters. It includes an initial call to `super`().
        5. ✓ Add a `String` parameter. Save. This should clear the compilation error.
        6. ✓ Use `String.split`() on `"\t"` to split the parameter up into tokens. If you are not sure how to split a string in Java, remember that **Google is your friend**.
        7. ✓ ~~Assign each token to the corresponding field~~. ~~Hints: You should trim leading and trailing whitespace before assigning the token to the field. This prevents any tabs or newlines from becoming part of a song property. Also: you'll need to convert a string to an integer for the `rank` field. If you are not sure how to do this, try googling "java convert string to integer example"~~.
    2. When you think you've got the constructor working properly:
        1. ✓ Open `SongListTester` in the editor.
        2. ✓ Uncomment `testNewSongFromTabDelimitedString`() in `SongListTester`. Be sure to also uncomment the `@Test` line before that function.
        3. ✓ Now Try running `SongListTester` again. If `testNewSongFromTabDelimitedString` doesn't pass, modify your new constructor to fix any problems.
8. ✓ In the next step, we'll build the `Main` class shown in the UML diagram above. It will have a method called `main`(), that is the starting point for our application. `main`()'s basic job is shown in the diagram below. Details are given in the steps after the diagram.

ROCK COUNTDOWN PROCESSING



9. `Main` class: 아키참아 하지만 마지막 스텝이니까 견디자.
    1. ✓ Make sure that your `Main` class has `import static sbcc.Core.*;` on a line near the top of the file.
    2. Add code to `main`() to do the following:
        1. ✓ Scan the ~~song list filename from the standard input~~ (don't print a prompt to the user). You can use `readLine()` or the **Scanner object's** `nextLine`() method to get the filename.
        2. ✓ Read the whole file into a `String` with `readFile`().
        3. ✓ **Split** the `String` on `"\r\n"` to get an array of lines from the file. `"\r\n"` is the end-of-line marker in Windows files. See **wikipedia** if you want a thorough coverage of end-of-line markers on various operating systems.
        4. ✓ Create a new `ArrayList<Song>`.
        5. ✓ For each line in the array (of lines from the file)
            1. ~~Create a song object from the line~~
            2. ~~Add the song object to the `ArrayList`~~.
        6. ✓ Create a **StringBuilder** that will hold the text we want to output.
        7. ✓ Iterate through the array list from the end to the start, appending: the song's rank, `"\t"`, the song's title, and `"\r\n"`.
        8. ✓ Write the `StringBuilder` to the standard output (use `println()`, not ~~System.out.println()~~). This will result in a blank line after the last song is printed.
        9. ✓ Write `"Complete"` to the standard output.
    3. Test your program manually:
        1. ✓ Download (right-click | Save Link As...) this **song list file.** You can use it to test your program.
        2. ✓ In the package explorer, right-click on your Main class | Run As | Java Application. In the Console view, type the name of the song list file (`rs30.txt`), then type <ENTER>.
        3. ✓ Your program should output the countdown (see the Sample Input and Output section below).

       한번 써봐.
10. Unit Test:
    1. ✓ Uncomment `testReverseList`(). This may cause some errors because the uncommented code makes calls to libaries that haven't been imported yet. You can type Ctrl-Shift-o to "organize the imports". This should fix the problems.
    2. ✓ Run `SongListTester` as a JUnit Test.
    3. ✓ If there are any problems, modify your code to address them and get `testReverseList`() to pass.

SAMPLE INPUT AND OUTPUT (input is blue, output is black)

```
rs30.txt
30	I Walk the Line
29	Help!
28	A Day in the Life
27	Layla
26	(Sittin on) the Dock of the Bay
25	God Only Knows
24	People Get Ready
23	In My Life
22	Be My Baby
21	Born to Run
20	Let It Be
19	Hound Dog
18	Maybellene
17	Purple Haze
16	I Want to Hold Your Hand
15	London Calling
14	Blowin' in the Wind
13	Yesterday
12	A Change Is Gonna Come
11	My Generation
10	What'd I Say
9	Smells Like Teen Spirit
8	Hey Jude
7	Johnny B Goode
6	Good Vibrations
5	Respect
4	What's Going On
3	Imagine
2	Satisfaction
1	Like a Rolling Stone

Complete
```

좋밥인정? 너인정!

SCORING

```
5 pts - testNewSongFromFields
5 pts - testNewSongFromTabDelimitedString
6 pts - testReverseList
4 pts - testUsesTemplateCorrectly
```