Objectives

The goal of this assignment is to enhance your understanding of `java.lang.String` and `java.util.Random` classes. Additionally, you will learn how to write conditional statements using `if/else` statements and boolean operators. Specifically, you will develop a Java program that allows the user to play different pieces of music. Finally, you will explore meaningful musical expression and learn how such expression can be created through Java programming environment.

Reading Assignment

To review `java.lang.String` and `java.util.Random` classes please check Chapter 3. To learn more about `if/else` statements, boolean and comparison expressions, review Sections 5.1–5.3 in your textbook. You should also read pages from the supplementary material on JFugue, that are given throughout this assignment, to learn about its functionalities, and for more information you can refer to the JFugue API.

Accessing the Laboratory Assignment on GitHub

This is your second team-based assignment. You have to work in groups of two or three for this assignment. You may select your own partner once again. Please let the instructor or a teaching assistant know if you would like help finding a team member. You will only need to deliver one submission from your team.

Once you have formed a team, you should designate a team leader - this will be the person accepting the assignment. Then, your team should fill out the Google Form given in the #labs channel of our Slack team. Finally, your team leader should find the announcement that provides a link for the assignment and accept it to see that GitHub Classroom created a new GitHub repository to access the assignment’s starting materials and to store the completed version of your assignment. Once your team leader’s repository is successfully created, the team leader should go to the “Settings” of the repository and add other member(s) of the team as “Collaborators”. Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, not completing the assignment in a team and not accepting the assignment means that you automatically receive a failing grade for it.

Before you move to the next step of this assignment, please make sure that you read all of the content on the web site for your new GitHub repository, paying close attention to the technical details about the commands that you will type and the output that your program must produce. Now you should clone the repository, ensuring that you have selected “Clone with SSH” option. For instance, if the course instructor ran the “git clone” command in the terminal for her team named ‘team1’, it would look like:

```
$ git clone git@github.com:Allegheny-Computer-Science-111-S2018/computer-science-111-spring-2018-lab-6-team1.git
```

After this command finishes, you can use “cd” to change into the newly created repository.

JFugue for Music Creation

Part of the richness of Java is the large number of libraries and APIs that are available for free. In this lab, we will use the music generator JFugue. JFugue is open source software, distributed under the LGPL license. This means the source code (written entirely in Java) is available for us to look at, modify, and adapt in our own programs as long as we credit the original authors and follow the other restrictions
on the LGPL license. For this assignment, we will not modify JFugue itself, but we will use the classes and methods it provides. A nice feature of JFugue is that it allows a user to submit a pattern - a string - with a fairly simple syntax, that is then converted to music.

JFugue’s features are fairly extensive, however we will only concentrate on a few classes and methods that can be used to create and play specific notes on various instruments. The main reason JFugue is easy to use is the MusicString, which is a specially formatted String object that contains musical instructions. For example, ‘‘C D E F G A B C’’ is a MusicString. In this example ‘‘C D E F G A B’’ corresponds to ‘‘Do Re Me Fa So La Ti’’ . You can change ‘‘C D E F G A B C’’ string to ‘‘E5s A5s C6s B5s E5s B5s D6s C6i E6i G#5i E6i | A5s E5s A5s C6s B5s E5s B5s D6s C6i A5i Ri’’ to play Bach’s Inventio 13. You can learn more about the MusicString style on page 41 of the supplemental handout.

Player class is responsible for defining and playing music in a Java program. Pattern an object that contains a MusicString. It allows you to create a piece of music that can be played using the Player class methods. The instrument by default is a piano. The supplementary document contains a list of all possible instruments on page 32. To choose a different instrument, you would include instrument token, I, followed by the instrument number in your pattern. For example, to play your notes as a flute, you need to create an instance of the Pattern class as follows:

```
Pattern pattern = new Pattern("I[Flute] C D E F G A B"); or as:
Pattern pattern = new Pattern("I73 C D E F G A B");
```

You can change the tempo of the music in JFugue. The tempo indicates how quickly a song should be played. It should often one of the first things set in a music String, since it applies to all musical notes that follow the tempo command. The default tempo is 120 beats per minute (roughly Allegro). If you wanted to play music slower, you could change the tempo to Lento, which corresponds to number 55 in JFugue as:

```
Pattern pattern = new Pattern("T[Lento] C D E F G A B"); or as:
Pattern pattern = new Pattern("T55 C D E F G A B");
```

Please refer to page 36 in the supplementary document to learn about different tempo settings.

If you are interested in learning more about JFugue, you are welcome to browse through the examples http://www.jfugue.org/examples.html, JFugue API and the supplementary document.

Music Creation Program

For this laboratory assignment you will write a program MusicGenerator that creates at least three different pieces of music and plays a random chosen piece according to user specified input. Before you begin creating your pieces of music, you need to decide on the theme of your music. Based on your theme, you should display a welcome message to the user. Additionally, your music pieces should be connected to your theme in some way. Your program should also read in user’s choice for an instrument and a tempo from the text file, and based on user’s choice, it should play the randomly chosen piece using the user-specified instrument with the specified tempo.

Specifically, your program should (not necessarily in the order listed here):

- Display all of team members’ names and the current date/time.
- Display a welcome message to the user with the theme of your music.
- Create at least three pieces of music, each lasting at least 30 seconds. Each of your music pieces should consist of at least two Music Strings, which can repeat.
- Randomly select one of your three pieces of music to play.
- Read in and save “instrument” and “tempo” as user’s inputs from the text file.
- Play the randomly selected musical piece using user’s preferred instrument and tempo.
Welcome to Happy Tunes!

Your selected instrument is a cello.
Your selected temp is Presto.
Playing music selected specifically for you ...
We hope you enjoyed this musical piece!

Testing your program

Your should build and run your program after you write every few lines of code. For example, you can build and run your program after you complete implementation of each step. You should also test your program using different types of instruments and tempo.

If you want to “build” your program you can type the command “gradle build” in your terminal, thereby causing the Java compiler to check your program for errors and get it ready to run. If you get any error messages, go back into gvim and try to figure out what you mis-typed and fix it. Once you have solved the problem, make a note of the error and the solution for resolving it. Re-save your program and then build it again by re-running the “gradle build”. If you cannot build MusicGenerator correctly, then please talk with a teaching assistant or the instructor.

When all of the errors are eliminated, you can run your program by typing “gradle run” in the terminal window—this is the “execute” step that will run your program and produce the designated output. You should test your program with different types of user input.

Technical Writing

After you have completed the programming tasks for the assignment, please reflect on the process. What step did you find to be the most challenging? Why? You should write your reflections in a file, called writing/reflection.md, that uses the Markdown writing language. To complete this aspect of the assignment, you should write three high-quality paragraphs that contain no writing errors. The first paragraph should report on your experiences with the various commands and Java code segments, and it should also describe your strategy for this lab’s implementation. The second paragraph should outline all of the challenges you faced with this lab and your solution for overcoming these challenges. Finally, a third paragraph should describe each team member’s contribution. Please explain what strategy you took as a team to complete this assignment (e.g., whether you used pairwise programming technique, if you divided the work in a certain way, how frequently you met outside of the lab session, etc.).

Checking the Correctness of Your Program and Writing

As verified by Checkstyle, the code for the src/main/java/labsix/MusicGenerator.java file must adhere to all of the requirements in the Google Java Style Guide available at https://google.github.io/styleguide/javaguide.html. The Markdown file that contains your reflection must adhere to the standards described in the Markdown Syntax Guide https://guides.github.com/features/mastering-markdown/. Instead of requiring you to manually check that your deliverables adhere to these industry-accepted standards, the GatorGrader tool that you will use in this laboratory assignment makes it easy for you to automatically check if your submission meets these well-established standards for correctness. Please see the instructor if you have questions about GatorGrader.

To get started with the use of GatorGrader, type the command “./gatorgrader.sh --start” in your terminal window. Once this step completes you can type “./gatorgrader.sh --check”. If your work does not meet the assignment’s requirements, then you will see the following output in your terminal: “Overall, are there any mistakes in the assignment? Yes”. If you do have mistakes in your assignment, then
you will need to review GatorGrader’s output, find the mistake, and try to fix it. Once your program is building correctly, fulfilling at least some of the assignment’s requirements, you should transfer your files to GitHub using the “git commit” and “git push” commands. For example, if you want to signal that the src/main/java/labsix/MusicGenerator.java file has been changed and is ready for transfer to GitHub you would first type “git commit src/main/java/labsix/MusicGenerator.java -m "Your descriptive commit message"” in your terminal, followed by typing “git push”, and then checking to see that the transfer to GitHub is successful. Remember, to correctly complete this assignment you need to commit all code and writing files to GitHub.

After the course instructor enables “continuous integration” with a system called Travis CI, when you use the “git push” command to transfer your source code to your GitHub repository, Travis CI will initialize a “build” of your assignment, checking to see if it meets all of the requirements. If both your source code and writing meet the established requirements, then you will see a green ✓ in the listing of commits in GitHub after awhile. If your submission does not meet the requirements, a red ✗ will appear instead. The instructor will reduce a student’s grade for this assignment if the red ✗ appears on the last commit in GitHub immediately before the assignment’s due date. Yet, if the green ✓ appears on the last commit in your GitHub repository, then you satisfied all of the main checks, thereby allowing the course instructor to evaluate other aspects of your source code and writing. Unless you provide the instructor with documentation of the extenuating circumstances that you are facing, no late work will be considered towards your grade for this laboratory assignment.

Summary of the Required Deliverables

This assignment invites you to submit, using GitHub, the following deliverables. Both your reflection document and your src/main/java/labsix/MusicGenerator.java needs to contain all of your team members’ names.


2. A complete and correct version of src/main/java/labsix/MusicGenerator.java that both meets all of the established requirements and produces the desired textual output in the terminal. See Section named “Program Requirements” in this document for details of this requirement.

Evaluation of Your Laboratory Assignment

Using a comment that the instructor shares with you through the commit log in GitHub, you will privately receive a feedback on this assignment. Your grade for the assignment will be a function of whether or not it was submitted in a timely fashion, if your program received a green ✓ indicating that it met all of the requirements, whether your program correctly implemented the steps outlined in this assignment, and if your writing satisfied the stated requirement and contains no mistakes. In addition to studying the efficiency and effectiveness of your Java source code, the instructor will also evaluate the accuracy of both your technical writing and the comments in your source code. If your program or your writing do not meet the specified expectations, your grade will be reduced. Additionally, if your submission receives a red ✗, the instructor will reduce your grade for the assignment while still considering the regularity with which you committed to your GitHub repository and the overall quality of your partially completed work. Please see the instructor if you have questions about the evaluation of this laboratory assignment.

Suggestions for Success

• Use the laboratory computers. The computers in this laboratory feature specialized software for completing this course’s laboratory and practical assignments. If it is necessary for you to work on a different machine, be sure to regularly transfer your work to a laboratory machine so that you
can check its correctness. If you cannot use a laboratory computer and you need help with the configuration of your own laptop, then please carefully explain its setup to a teaching assistant or the course instructor when you are asking questions.

- **Follow each step carefully.** Carefully read each sentence in the assignment sheet, making sure that you precisely follow each instruction. Take notes about each step that you attempt, recording your questions and ideas and the challenges that you faced. If you are stuck, then please tell a teaching assistant or instructor what assignment step you recently completed.

- **Regularly ask and answer questions.** Please log into Slack at the start of a laboratory or practical session and then join the appropriate channel. If you have a question about one of the steps in an assignment, then you can post it to the designated channel. Or, you can ask a student sitting next to you or talk with a teaching assistant or the course instructor.

- **Store your files in GitHub.** As in the past laboratory assignments, you will be responsible for storing all of your files (e.g., Java source code and Markdown-based writing) in a Git repository using GitHub Classroom. Please verify that you have saved your source code in your Git repository by using “git status” to ensure that everything is updated. You can see if your assignment submission meets the established correctness requirements by using the provided checking tools on your local computer and by checking the commits in GitHub.

- **Keep all of your files.** Don’t delete your programs, output files, and written reports after you submit them through GitHub; you will need them again when you study for the quizzes and examinations and work on the other laboratory, practical, and final project assignments.

- **Back up your files regularly.** All of your files are regularly backed-up to the servers in the Department of Computer Science and, if you commit your files regularly, stored on GitHub. However, you may want to use a flash drive, Google Drive, or your favorite backup method to keep an extra copy of your files on reserve. In the event of any type of system failure, you are responsible for ensuring that you have access to a recent backup copy of all your files.

- **Explore teamwork and technologies.** While certain aspects of the laboratory assignments will be challenging for you, each part is designed to give you the opportunity to learn both fundamental concepts in the field of computer science and explore advanced technologies that are commonly employed at a wide variety of companies. To explore and develop new ideas, you should regularly communicate with your team and/or the teaching assistants and tutors.

- **Hone your technical writing skills.** Computer science assignments require you to write technical documentation and descriptions of your experiences when completing each task. Take extra care to ensure that your writing is interesting and both grammatically and technically correct, remembering that computer scientists must effectively communicate and collaborate with their team members and the tutors, teaching assistants, and course instructor.

- **Review the Honor Code on the syllabus.** While you may discuss your assignments with others, copying source code or writing is a violation of Allegheny College’s Honor Code.