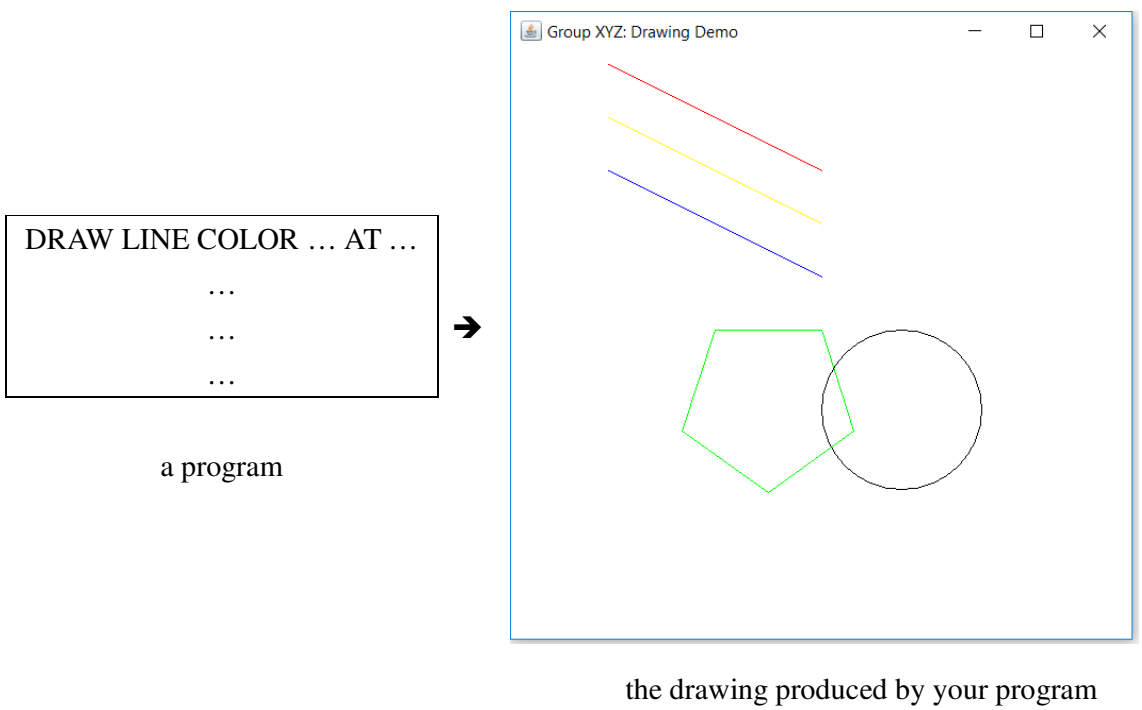
COMP3021 Programming Languages Paradigm (Spring 2021)

Group project: **Programming Language for Drawing**
Group size: 2-3 students per group

1. Overview

Several programming languages can be used to draw pictures. For example, some general-purpose languages (e.g., **C++** and **Java**) support drawing functions by calling library functions. On the other hand, specialized programming languages (e.g., **Logo**) contain built-in keywords and syntax for drawing.

In this project, your group will **design and implement** a new programming language to draw pictures. Figure 1 shows an example of program and the drawing produced by your program. In addition, you will need to highlight the features of your programming language and **convince** users why they should use your programming language (instead of existing programming languages).



a program

the drawing produced by your program

Figure 1: Example of your programming language

2. The Compilation Process

Figure 2 shows the compilation process for this project. It has three layers.

Layer 1: Design a grammar file for your programming language;
Use SableCC to compile them into compiler classes

Layer 2: Modify compiler classes to implement semantic analysis and code generation;
Compile a drawing X (written in your language) into a Java drawing class

Layer 3: Compile the Java drawing class with the "Draw Engine"

Note that the components shown in blue color are given to you (e.g., the "**Draw Engine**" written in Java).

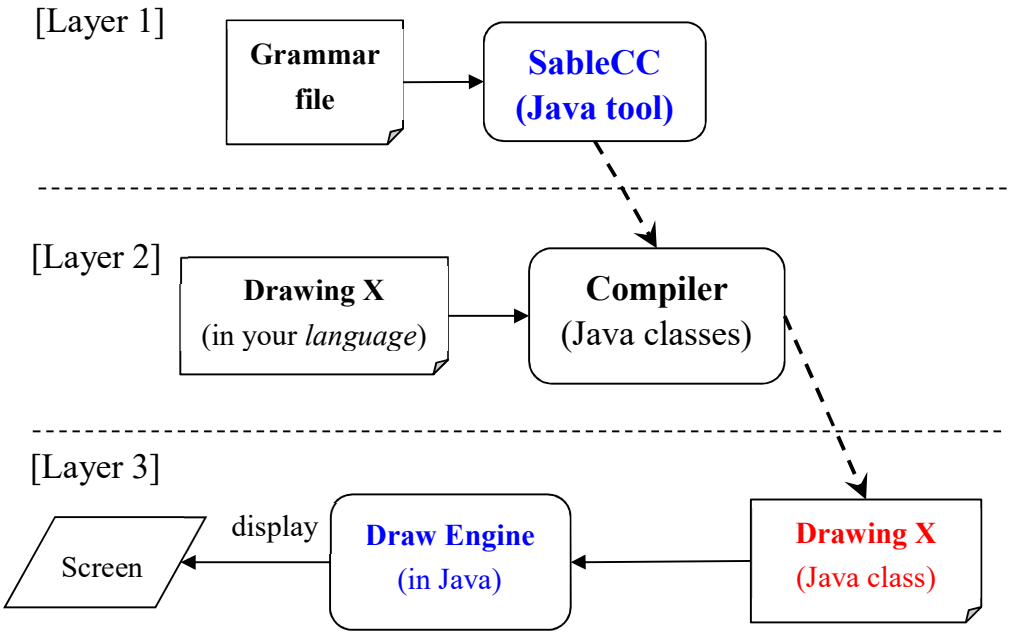


Figure 6: Compilation process

Your task is to develop the components shown in black:

(i) your **programming language** for writing any drawing X
   a. Design a grammar file (with lexical rules & syntax rules)

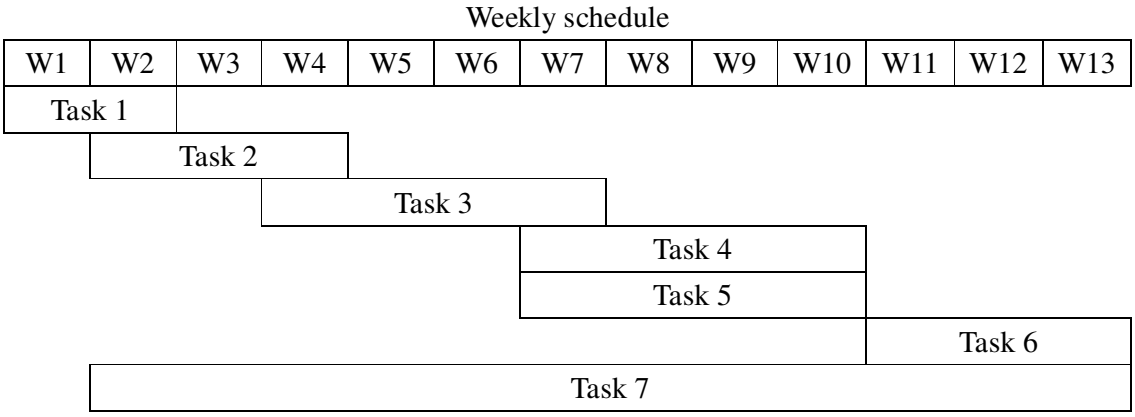
(ii) the **compiler** for converting the drawing X into a **Java class** (in red color) such that it can be integrated with the "Draw Engine"
   a. Convert the grammar file into compiler classes [by using SableCC]
   b. Modify compiler classes to implement semantic analysis and code generation

3. Tasks and Schedule for the Project

In this project, you should form a group of **2-3** students to design and implement the above programming language. Your group needs to submit a **progress report**, a final **project report** (and source codes) and a **demo video** at the end of this course. Throughout the course, we will guide you on essential knowledge, skills, and tools for the project.
Here are the deadlines of your deliverables:

Deliverable	Deadline
Group list (in Blackboard)	30 January 2021
Progress report	20 March 2021
Final report & demo video	24 April 2021

You are recommended to use "iterative and incremental development" in this project.
Here is the list of tasks and schedule for your group.



Task 1. Form groups

- Find your classmates to form a group (by using Teams, Whatsapp, Wechat)
- Elect the group leader and discuss the role of each member
- Submit your group list in *Blackboard*

Task 2. Analyze the problem and requirements

- Compare **existing programming languages** on drawing by using language evaluation criteria (e.g., readability, writability, reliability)
- Identify the main requirements and features of your programming language

Task 3. Design the language

- Use your language to write a few small sample programs
- Design the data types supported by your language
- Design the syntax (i.e., grammar) of your language
- Design the semantics of your language (e.g., attribute grammar, scope rules, typing)
- Check your sample program with the syntax and semantics of your language; revise your language if necessary

Task 4. Implement the language

- Use a compiler tool (SableCC)
- Select the **core features** of your language to be implemented
- Prepare the grammar file (with lexical rules & syntax rules)
- Modify compiler classes to implement semantic analysis and code generation

Task 5. Testing

- This task proceeds **in parallel** with the task 4
- Design test cases (several sample programs written in your language)
- Test whether your sample programs can be compiled successfully
- Test whether your drawing classes work correctly

Task 6. Iterative development

- Your group should have completed the previous iteration of tasks 4 and 5
- Decide the next important feature to be implemented in the current iteration
- Add this feature to your language by repeating tasks 4 and 5
- Repeat this development process until the compiler classes become ready

Task 7. Write report

- This task proceeds **in parallel** with all the tasks shown above

  Don't start writing the report at the end of this semester! Start early!
- Each time when you complete a task (e.g., analyzing requirements), try to summarize what you have done in a document
- Circulate the document among your group members for their comments
- Refine the chapters and combine them into a report at the end of semester

4. Important Deadlines

4.1 Group Formation

Deadline: 30 January 2021

Try to meet with your classmates (either face-to-face or online).

Remember that your group should contain **2-3 students**.

Please elect a **group leader** to contact with us in future.

If you have difficulty in group formation, please *email to us*. We will assign you to a group.

4.2 Progress Report

Deadline: 20 March 2021

The weighting of this part is **10%** of project marks.

Your group leader will have to submit the progress report as a single **pdf** file in *Blackboard*.

In order to keep track of your progress, your group will submit a **progress report** (about 5-10 pages, single-spacing, font size 12). The progress report should have these sections:

1. A Simple Introduction
2. Requirement / Problem Analysis
   - Summarize what you have done in task 2
   - Show the main requirements of your programming language
3. Design
   - Summarize what you have done in task 3
   - Show the data types, syntax, and semantics of your language (designed so far)
   - Show some sample programs written in your language

4.3. Project Report

Deadline: 24 April 2021

The weighting of this part is **50%** of project marks.

The **group leader** will have to submit the followings in *Blackboard*:

(i) project report as a single **pdf** file, and

(ii) a single **zip** file that contains your source codes, including (1) a grammar file, (2) .java and .class files, (3) sample input files and output screenshots of results, (4) a README file about how to compile and run your codes.

Make sure that your source codes can be compiled and executed properly.

Please be reminded that YOUR project report must be written in **your own words** about what you have learnt from the topic. Try to use your own (textual or graphical) examples to illustrate your ideas and methods. ***Any quoted materials must be appropriately cited.***

Report format

Use single-spacing, and the "Times New Roman" font with font size 12.

Your report should contain the following sections:

0. Contribution statement
    - For each member, state his/her contributions in tasks 2-7
    - For each member, state which sections in this report are written by him/her
1. Introduction
2. Requirement & Problem Analysis
    - Document what you have done in task 2
3. Design
    - Document what you have done in task 3
4. Implementation
    - Document what you have done in task 4 (including the work in task 6)
5. Testing
    - Document what you have done in task 5 (including the work in task 6)
6. Conclusion
7. Appendix

Page limit

The total page limit of Sections 1—6 is 30 pages.

You are recommended to put important information in Sections 1—6.

There is no page limit for the appendix (Section 7).

4.4. Demo video

Deadline: 24 April 2021

The weighting of this part is **40%** of project marks.

Each group should prepare a 10-minute demo video of their sample programs and drawings.

Details will be announced later.

5. Grading for the project

Grade	*Language design (in report)*	*Examples (in both report and demo video)*	*Presentation (in both report and demo video)*	*Requirement analysis (in report)*	*Testing (in report)*
A+/A/A–: outstanding	Novel features, easy to read & write, reliable	Abundant examples to illustrate your features	Perfect presentation	In-depth analysis	Detailed testing of the compiler
B+/B/B–: very good / good	Reasonable features, easy to read & write	Reasonable number of examples to illustrate your features	Good presentation	Reasonable analysis	Reasonable testing of the compiler
C+/C/C–: Satisfactory	Basic features with minor flaws	Few examples to illustrate your features	Presentation with some unclear points	Mediocre analysis	Some testing
D+/ D: barely satisfactory	Basic features with major flaws	Unclear examples	Barely clear presentation	Limited analysis	Limited testing
F: Inadequate	Plagiarism	Incorrect examples	Totally unclear presentation or plagiarism	No analysis	No testing