The Practice of Teaching Java Programming Language to Undergraduate Students

Weijun Chen, Tsinghua University, China

The Asian Conference on Education 2021
Official Conference Proceedings

Abstract
The paper presents our practice of teaching java programming language to undergraduate students at Tsinghua University, China. The biggest challenge is the design of the course to improve students' programming ability.The course consists of two parts: the class lectures and the after-class exercises, both are designed deliberately. The purpose of the lectures is to help the students understand the kernel ideas of the object-oriented programming (OOP) and learn the common used java classes. OOP is difficult for many students, therefore we used different methods to make the process smoother. For example, Tony Stark and his armor in the movie "Iron Man" are presented when interpreting the storage of the superclass object and the subclass object. The java class library is an important component of the language which contains exceptions, input and output streams, files, graphics user interface, threads, network programming, etc.The most important part of the after-class exercises is a programming project. Learning by doing is probably the best way to get familiar with a new language. The students are required to write a complete program (generally a computer game) using java individually. In such a project, all the technologies learnt in classroom are used and combined together. The games written include Tetris, Snake, Battle City, etc.This course has been taught for 6 years at Tsinghua and about 740 students were enrolled totally. These students were from different majors. The examination results and the programs they wrote indicate that they achieved major improvements after the course.

Keywords: Java Programming, Course Design, Ability Development

Introduction

Java is one of the world's most popular programming language, it is widely used in software industry. For example, Elasticsearch is a real-time distributed search and analytics engine. It allows you to explore your data at a speed and at a scale never before possible. Many famous websites such as Wikipedia, StackOverflow, Github, Facebook, Quora, LinkedIn, Netflix are using Elasticsearch to search all kinds of documents, and this powerful search engine is developed in Java.

We provide a Java programming course for undergraduate students at Tsinghua University, China. It is an optional course for anyone interested in studying Java programming. The students are required to have basic prior programming experience before taking this course. For example, they know how to write a procedure to solve a practical problem in the C language. The course is given in one semester of sixteen weeks, and in each week, there is one three-hour lecture and one three-hour lab session. Generally there are more than 70 students in the class, they are from different majors such as materials science and engineering, chemical engineering, hydraulic engineering, mechanical engineering, automotive engineering and building science & technology. The textbook used is Introduction to Java Programming, which was written specifically for this course by the author of this paper.

The course consists of two parts: the class lectures and the after-class exercises, both of which are important in learning Java language. The class lectures tell the students the basic ideas of the object-oriented programming and help them to learn the common used Java classes. The after-class exercises give students the opportunity to practice the theory and skills learned in the classroom. Learning by doing is particular important when you try to study a new programming language.

The author of the paper (Shi, 2012) have taught the course for several years and find that it is not an easy job. Firstly, the students are from different majors and they have few (if not zero) professional training before taking the course. For example, the code they write may be correct functionally, however they are in bad programming styles and can hardly be understood by other programmers. This situation can lead to serious problems when many programmers cooperate with each others to complete a software project.

Secondly, the course is an optional one for the students and they may not spend as much time on it as their major courses. A programming course is always tough enough because it has a lot of homework. The students need to write source code on computer and any small mistakes will make the program fail to run correctly. Therefore, it will take a student lots of time to complete the course. However, many students have their major courses which are also tough enough at the same semester and it is obvious that these courses will have priority when time is limited.

Thirdly, the students enrolled in this course may have learnt how to write programs in structural programming languages (such as the C language), but few of them have any idea of the concept of the object-oriented programming. They are often confused by the difficult OOP concepts such as Abstract, Encapsulation, Inheritance and Polymorphism. A common situation is that a student can write a function to solve a particular problem, but he doesn't know how and why to combine several functions with a couple of variables to make a class.

Therefore, it is a great challenge for us to provide better service to the students when teaching this course, we need to design the course carefully and elaborately.

Class lectures

The purpose of the class lectures is to help the students understand the kernel ideas of the object-oriented programming and learn the common used java classes.

Topics covered in the course include:

- Introduction: history of Java language, the Java platform, structure of a Java program, Java IDE (Integrated Development Environment)
- Java language basics: variables, constants, data types, expressions and operators, the if statement, the switch statement, the for loop statement, the while loop, the do-while loop, the break and continue statements, one-dimensional arrays, two-dimensional arrays
- Java OOP: classes and objects (defining classes, using classes, methods, references, static types), access control (public, protected, default, private), method overload, storage management, inheritance (parent and subclass), polymorphism (override, method binding, abstract classes, interfaces)
- Exceptions: what is an exception, why use an exception, try-catch structure, throws
- Input and output: byte streams, char streams, buffers, files
- Graphic User Interface (GUI): the Graphics class, the Color class, the Font class, Swing components (containers, basic controls, layout manager, event handling)
- Thread: process and thread, Java thread, data sharing between threads, thread mutex and synchronization, thread scheduling and priority
- Network programming: principles of computer networks, URL(Uniform Resource Locator), TCP(Transmission Control Protocol) programming, UDP(User Datagram Protocol) programing
- Object collections: collections, List interface (ArrayList, LinkedList), generic types, Set interface (HashSet, TreeSet), Map interface (HashMap, TreeMap)
- Android programing: mobile application development, the Android operating system, the Android IDE, develop an Android application
- Writing solid code: coding conventions (naming, comments, error handling, style and layout), program debugging, test cases development

These topics can be divided into three parts. The first part discusses the Java language basics, i.e. how to write a computer program in Java language. As we know, when James Gosling and his colleagues invented the Java language 30 years ago, they thoroughly studied and compared the C/C++ language, therefore, this part of the Java language is quite similar to that of C/C++.

The second part is Java OOP, which is the core foundation of the course. Every Java application is a combination and interaction of different kinds of classes. Furthermore, every programmer need to use the libraries implemented by the language system when developing a practical software project and the libraries provided by Java are also in the form of classes. However, students often have difficulty in learning the OOP part of the Java because it is abstract and hard to understand. Therefore, we have to try our best to teach the students using all kinds of valid methods.

The third part is commonly used Java class libraries. To become an experienced Java programmer, it is very important to get familiar with these libraries. We don't have to build a software system from scratch, we can build it on top of the existing components.

The methods

To make students understand the contents of the class lectures easier, various of methods are used.

The first method is to propose analogues in life for those abstract concepts in theory. When students see those difficult concepts, they may feel that it is hard to understand. But when they see these visual examples, they know what it means immediately.

For example, in class we will introduce the memory storage of objects. Given a superclass and a subclass, if we construct a subclass object, on the one hand this object is an independent and complete object, on the other hand there is also a superclass subobject hidden inside it. When the students read this paragraph, generally they will get totally confused. Then we will show them the Figure 1.

Figure 1: Onion and Iron man.

In the Figure 1, we can see an onion. Onion is a vegetable that is composed of several layers. If you skin the outside layer of an onion, the remaining part is still a complete onion that has the same shape. Another example is the ironman from the film "Iron Man". Ironman is a superman that can fly and fight, actually he is a man named Tony Stark who is wearing a suit of armor.

```
class Man {
    String name;
}
class IronMan extends Man {
    String nickname;
    void print() {
        System.out.println("Man: " + name);
        System.out.println("Ironman: " + nickname);
    }
}
```

Figure 2: the Man and IronMan class.

In figure 2, we defined a Java superclass named Man and a derived subclass named IronMan. Now if we create an IronMan object, then there is also a Man object existing in it. Just like there is always a man (Tony Stark) existing in the ironman.

The second method is being humorous in class. In a certain sense, studying is hard and boring. If a student has listened to the lectures for a long time, he or she will fail to concentrate more on the work. Therefore, it will be helpful if we can attract the students' attention by saying something funny now and then. For example, logical operators are used to combine two boolean operands together and there are three logical operators: && (AND), || (OR) and ^ (XOR). Figure 3 demonstrates the truth tables of these operators.

a && b

a \ b	true	false
true	true	false
false	false	false

a || b

a \ b	true	false
true	true	true
false	true	false

a ^ b

a \ b	true	false
true	false	true
false	true	false

Figure 3: Truth tables of three logical operators.

It is obviously that these tables are boring and hard to remember, so we can show them in another way. Figure 4 is a picture we found on the Internet, it is named as "boolean hair logic", which means the result hair of A and B's logical operators. For example, B has a moustache while A hasn't, therefore, A && B doesn't have one.

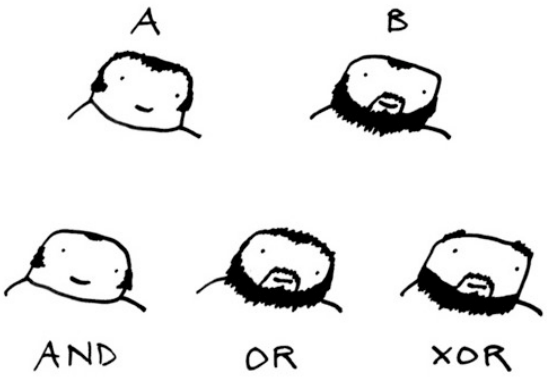


Figure 4: Boolean hair logic.

The exercises

As we know, learning by doing is probably the best way to get familiar with a new programming language. The after-class exercises of this course include two parts: weekly assignments and a programming project.

In each week, we will have a three-hour lecture that covers one of the course topics. After that, the students are required to finish weekly assignments which are generally two or three programming problems. The purpose of these assignments is to help the students review what they have learned in class.

The most important part of the after-class exercises is a programming project. The students are required to write a complete program (generally a computer game) using Java individually. In such a project, all the technologies learnt in classroom (classes, exceptions, GUI, multi-threading, network programming, etc.) are used and combined together.

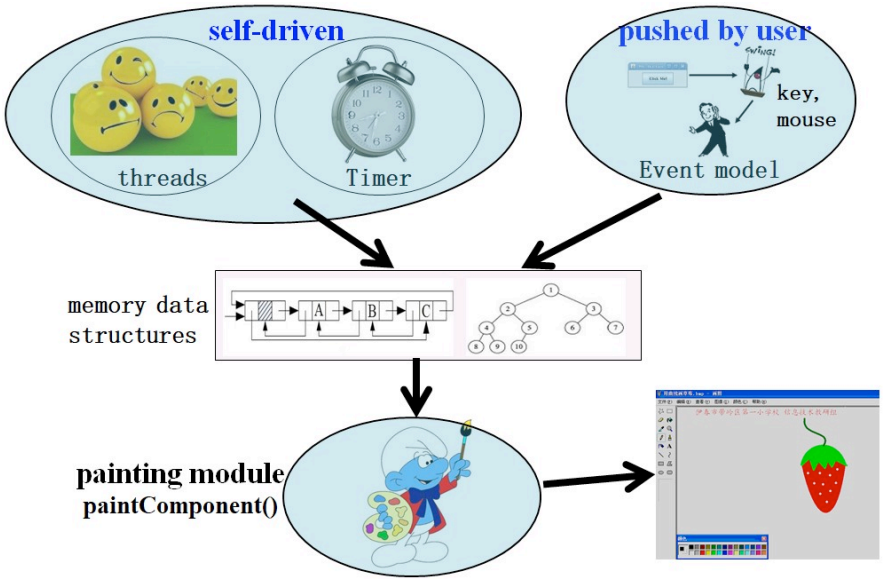
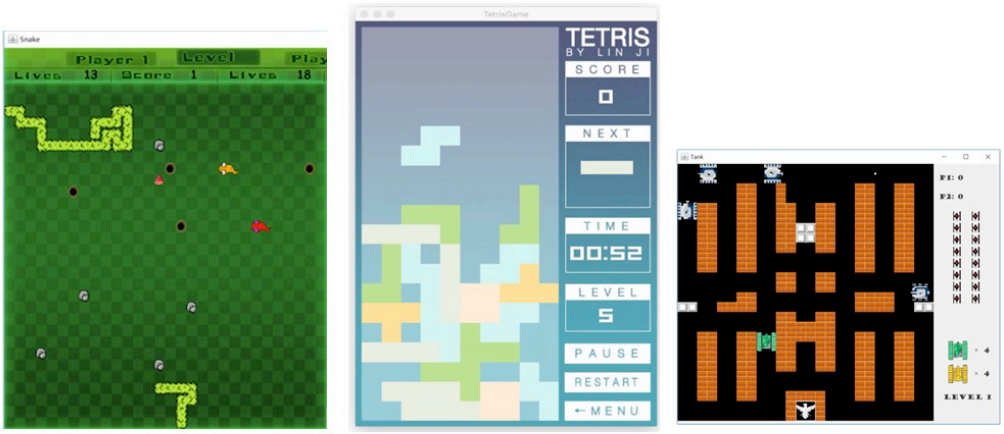


Figure 5: A general architecture of Java GUI games.

For a typical computer game, animation is the core foundation of the whole system. Lots of things are moving such as tanks, planes, guns, etc. However it is not easy to implement GUI animation especially for beginners. In Figure 5, we proposed a general architecture of Java GUI games, with which the students can realize any kind of GUI animation they want.

Typically there are two types of moving in a computer game, one is called self-driven, which means that type of object will move by itself. This can be implemented with Thread or Timer class in Java. The other moving is caused by user's movements such as pressing a keyboard key or a mouse button. This can be implemented with the event model. Both types of moving will lead to the value changes of the data structures in memory. And there is an independent painting module (generally the painComponent() function of the JPanel class) that is responsible for refreshing the screen.



Snake Tetris Battle City

Figure 6: Games written by students.

During the past semesters, the students have written different types of computer games such as Snake, Tetris, Battle City, etc., which is demonstrated in Figure 6.

Conclusion

We began to teach this course 6 years ago and about 720 students were enrolled totally. These students were from different majors and they only had basic programming experience before taking this course. The examination results and the programs they wrote indicate that they achieved major improvements after the course was over.

The grading policy of the course is: weekly assignments (30%), programming project (40%), final exam (25%) and class participation (5%).

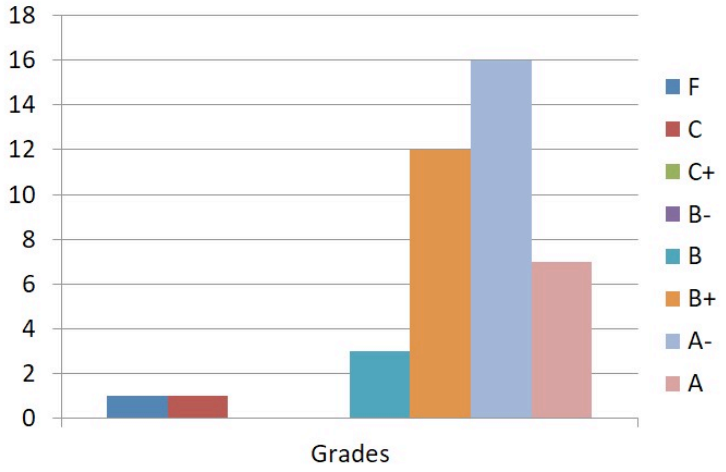


Figure 7: students' grades in the last semester.

Last semester, 40 students were enrolled in the course, and Figure 7 shows their final grades. 23 students (57%) got A or A- and 15 students (37%) got B+ or B. On the other hand, there was one student who failed the course. The reason is he didn't complete the programming project before the deadline.

References

Shi K. L., Chen W. J., Zhang L. and Luo L. J. (2012). Kaleidia: A Practical E-Learning Platform for Computer Programming Courses. In Proceedings of the Canada International Conference on Education (CICE-2012), June 18, 2012, University of Guelph, Canada.

Contact email: cwj@tsinghua.edu.cn