1 What is Java?

The most simple answer to this question would be: Java is a platform independant programming language. Although, this actually doesn't describe Java well. If you're intereseted in knowing how things work, continue reading this section, otherwise you can skip down to the next section.

Java is more comletely described as the Java Development Kit (JDK), a software platform made up of various components such as the JVM and JRE. This seems like a lot at first, but it actually isn't too bad. In order to start developing in Java one has to download the JDK which includes tools like the Java compiler/debugger and the Java Runtime Environment (JRE). The JRE is a piece of software that contains everything needed to run a compiled Java program (called a Java Bytecode - similar to a C++ binary) including the Java libraries, the Java Virtual Machine (JVM), and deployment tools such as browser plugins so your Java program can be run over networks and inside browsers without you having to worry about gritty low level implementation details. The JVM is basically a simulated CPU that can be installed on a multitude of platforms. This is how Java achieves platform agnosticism.

Another benefit of the Java runtime is that it takes care of memory management through a process called garbage collection (GC). This provides the develper the oppurtunity to focus on other aspects of program design and can prevent issues like memory leaks and memory based security exploits. This allows for the development complex systems with improved reliability and safety.

There are some drawbacks of large runtimes like the JRE. The most noteworthy being inconsistent performance. You don't know when the GC will run and if you are developing performance critical code, think airplanes or stoplights, then Java might not be the best tool for you. Java uses an incremental GC system so these inconsistencies are almost always negligible. Another issue is long start up time. The runtime is so large it takes a minute to get going. This issue has been quite the challenge for developers at Oracle to address. When you start using IntelliJ you will understand exactly what I mean and avoid closing the IDE at all costs. Lastly, if there is a bug in the runtime there is very little you can do to fix your software. Although, given the incredible complexity of modern compilers, similar issue exist within languages that don't provide runtimes.

2 Why use an IDE now?

IDE's contain tools that are incredibly useful for developing large projects by providing tools like program frameworks, continuous integration systems, and static or dynamic analysis. You won't be using any of these tools for some time but it is important to start getting used to different development environments that you may have to use throughout your education and careers. It is important to have a strong base of knowledge before utilizing these tools as to not become too dependent on them. Check out the nearly 4000 plugins available for IntelliJ from their plugin repository and you will likely find something that will contribute to your workflow. One plugin that is particulary useful is the Vim plugin, which updates the key bindings and provides the same multi-mode interface. It even allows you to source a vimrc for further customization.

3 Vocabulary

Java developers use some different vocabulary that is worth being aware of:

- Member: A Method or a Field of a class.
- Methods: Are functions of a class. If you use the words "member function" to a Java developer they might laugh at you.
- Fields: A piece of data in a class (like an int or a String). These are sometimes called variables as well but a variable also refers to local data where a field refers to a class member.

- Referance: Its a really fancy pointer. It is very different from a C++ referance and much closer to a C++ pointer. (a C++ referance is immutable and CANNOT be null)
- final: A Java keyword that designates something cannot be changed. A final class cannot be derived from, a final variable is like a C++ const variable, and a final method cannot be overridden.
- static: A Java keyword that is the same as C++ but is worth noting. A static field is shared by all instances of the class and a static method is called using the class itself and not an instance of the class. Static methods can be invoked even when no objects of that class have been instantiated and can only operate on static class members. Static methods and variables are also often called class methods and class variables.
- super: A Java keyword to access members of a class's parent.
- Supertype (of type A): Any class or interface above A in the inheritence tree.
- this: A Java keyword to represent the instance of the class in which it appears. Although Java does not require, it is idiomatic in Java to use this.member whenever you are accessing members within a class.
- extends: A Java keyword to declare that a class is a subclass of another.
- interface: A Java keyword used to define a collection of methods and constant values. Simlar to an abstract base class with some critical differences. Interfaces can be used as types and are heavily utilized in the Java Collections.

4 Tips for Success

Use Oracle's documentation. The Java version at time of writing is Java SE 13 with the most recent LTS version being Java 8. You probably downloaded the the SE 13 version of the JDK. The API documentation for SE 13 is found here:

https://docs.oracle.com/en/java/javase/13/docs/api/index.html

Oracle's documentation is incredibly detailed and organized. Any question you have about the language can be answered by reading them. In the process of looking for the answer you will learn things about the language and develop skills that are not limited to Java. The tutorials page in the documentation is also full of easy to read and detailed explanations of various topics. Stack Overflow and other tutorial websites can be useful to gather information but be careful to assess the quality. Try using these resources to find where in the documentation to look instead of massaging other's code examples.

In extension to the previous piece of advice, consider using the Java collections. Program 5 details that you have to construct some of your own data structures from scratch but you can use the libraries for anything else you would like. I suggest looking at the ArrayList and HashMap classes.

5 Using IntelliJ Idea

Copy in this section from old lab book, these steps seem mostly the same and is useful information.