Week 1
Great Software Begins Here

Packages

A **package** is simply a container that groups related types (Java classes, interfaces, enumerations, and annotations)

Map, Queue, Set, ArrayList, etc. belong to the java.util.* class.

packages help you to reserve the class namespace and create a maintainable code

Two types:

- Built-in Package
- User defined packages

Packages

How to define a Java package?

To define a package in Java, you use the **keyword** `package`

Java uses **file system directories** to store packages

The package name must be unique (like a domain name), but in reverse order. For example: edu.sjsu.cs

Use keyword `import`, to import packages in java

```
import package.name.ClassName; // To import a certain class only

import package.name.*
```

Java source file structure:

- A java Program can have multiple classes in same file but only one of which can be public.
- That public class will also be the file name of the class. Otherwise, compile time error.
- If there is no public class then we can name the .java file whatever we want.
- No relation between file name and main function in class.
- If a .java file has multiple classes, for every class a separate .class will be created.

Import Statements

```
class Test {
        public static void main(String args[]) {
                ArrayList l = new ArrayList();
        }
}
```

What's wrong with this?

Import Statements

- We can resolve this problem by using fully qualified name "java.util.ArrayList l=new java.util.ArrayList();".
- But problem with using fully qualified name every time is it increases length of the code and reduces readability.
- We can resolve this problem by using import statements. - import java.util.ArrayList;
- whenever we are using import statement it is not require to use fully qualified names we can use short names directly. This approach decreases length of the code and improves readability.

Import Statements

There are 2 types of import statements.

1) Explicit class import

    Example: Import java.util.ArrayList

    This type of import is highly recommended to use because it improves readability of the code.

2) Implicit class import.

    Example: import java.util.*;

    It is never recommended to use because it reduces readability of the code.

Exercise

https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

For the code below, what all imports should work:

```
public class PatternExample {
        public static void main(String args[]) {
                Pattern p = Pattern.compile("a*b");
        }
}
```

1) import java.*;
2) import java.util.*;
3) import java.util.regex.*;
4) import java.util.regex.Pattern;

Import Statements (Continued)

- Whenever we are importing a package all classes and interfaces present in that package are by default available but not sub package classes.
- In any java Program the following 2 packages are not require to import because these are available by default to every java Program.

  1. java.lang package

  2. default package(current working directory)

- "Import statement is totally compile time concept" if more no of imports are there then more will be the compile time but there is "no change in execution time".

Java Simple Hello World Program

Java's main function

public – Java's main function requires a public **access modified**.

static – Java's main method is static, which means no instances need to be created beforehand to invoke it.

void – Some programming languages return a zero or 1 to indicate the main method has run successfully to complete. Java's main function is void, which means it does not return any value when it completes.

main – When the JVM starts a standalone application, the main method is the function that gets invoked.

String[ ] – An array of configuration parameters to be used by the application can be passed into the main function as arguments.

args – The configuration parameters passed into the main function in Java are typically named args.

Access Modifiers

Access modifiers are used to set the accessibility (visibility) of classes, interfaces, variables, methods, constructors, data members, and the setter methods.

Modifier	Description
Default	declarations are visible only within the package (package private)
Private	declarations are visible within the class only
Protected	declarations are visible within the package or all subclasses
Public	declarations are visible everywhere