JAVA PACKAGES

CS302-Introduction to Programming
University of Wisconsin – Madison
Lecture 26

By Matthew Bernstein – matthewb@cs.wisc.edu

What are packages?

- According to Java:

> "A *package* is a grouping of related types providing access protection and name space management. Note that *types* refers to classes, interfaces, enumerations, and annotation types."

What does this definition mean?

- A **type** is any data type that can be stored in a variable in Java. So far, we have talked about two types: primitive types and class types.
- There are a few other kinds of types: interfaces, enumerations, and annotation types.
- In this course, we will talk about interfaces (at a later time).

Packages Group Types

- As you are very familiar with at this point, we define our own data types by implementing classes. For example, if we implement a Car class, we can now instantiate Car objects (their data type is of the class Car).
- In Java, it is good practice to group these classes (and other type definitions) into packages.
- On your computer, each package will actually get its own directory that will store the .java files that implement these types

Packages Allow Different Types With the Same Name

- Let's say we want to implement our own class called ArrayList
- As you know, ArrayList, is an existing class. Can we create a class with this same name?
- Yes! As long as our ArrayList class is in a different package from the other one, it is fine (the standard ArrayList is in the java.util package

Packages Define Namespaces

- A **Namespace** is a term you will hear a lot in programming.
- A namespace refers to a place where all data types must have unique names.
- Each package is thus a namespace because within a package you cannot have two types with the same name
- HOWEVER, we can have types with the same name in different packages

Using Types with Same Name in A Program

- Let's say, we create our own ArrayList class in a package named myprogram.data
- Now let's say in our main method we want to utilize both our own implementation of ArrayList AND the ArrayList implemented in the java.util package
- How do we differentiate the two?
- Answer: We must prefix the data type by the full package

Differentiating Data Types With Same Name

- Given the preceding scenario, we would do the following:

```
public static void main(String[] args)
{
        // Our ArrayList implementation
        myprogram.data.ArrayList a1 = new myprogram.data.ArrayList();

        // The standard ArrayList implementation
        java.util.ArrayList<Integer> a2 = new java.util.ArrayList<Integer>();
}
```

Import Statements

- Import statements allow us to forego prefixing our types with the package
- So for example, if we have the following import statement:

```
import java.util.ArrayList;
```

- Then every time we use the ArrayList type, it will automatically be the ArrayList found in java.util
- If we want to use our implementation of the ArrayList, we would need to prefix it with the package myprogram.data

Colliding Import Statements

- We cannot import two data types with the same name
- For example we CANNOT do the following:

```
import java.util.ArrayList;
import myprogram.data.ArrayList;
```

- If we have the following declaration:

```
ArrayList a;
```

- Our program will not know which ArrayList to use! Thus, these import statements will cause a compile-time error.