Organizing Java Classes

Up until this point in the class, we haven't created a lot of classes, so we haven't had to think much about organizing those classes.

As the course progresses, we're going to be using more and more of Java's libraries, and our applications are going to get more complex.

This feels like a good time to talk about the package and import statements, in more detail.

We've talked briefly about import statements, when we used the Scanner class, and we mentioned packages when we talked about access modifiers.

In this video, I want to focus on what a package is, why we'll be switching to using it from this period forward, and how to access classes in different packages.

package

As per the Oracle Java Documentation:

A package is a namespace that organizes a set of related types.

In general, a package corresponds to a folder or directory, on the operating system, but this isn't a requirement.

When using an IDE, like IntelliJ, we don't have to worry about how packages and classes are stored on the file system.

package

The package structure is hierarchical, meaning you group types in a tree fashion.

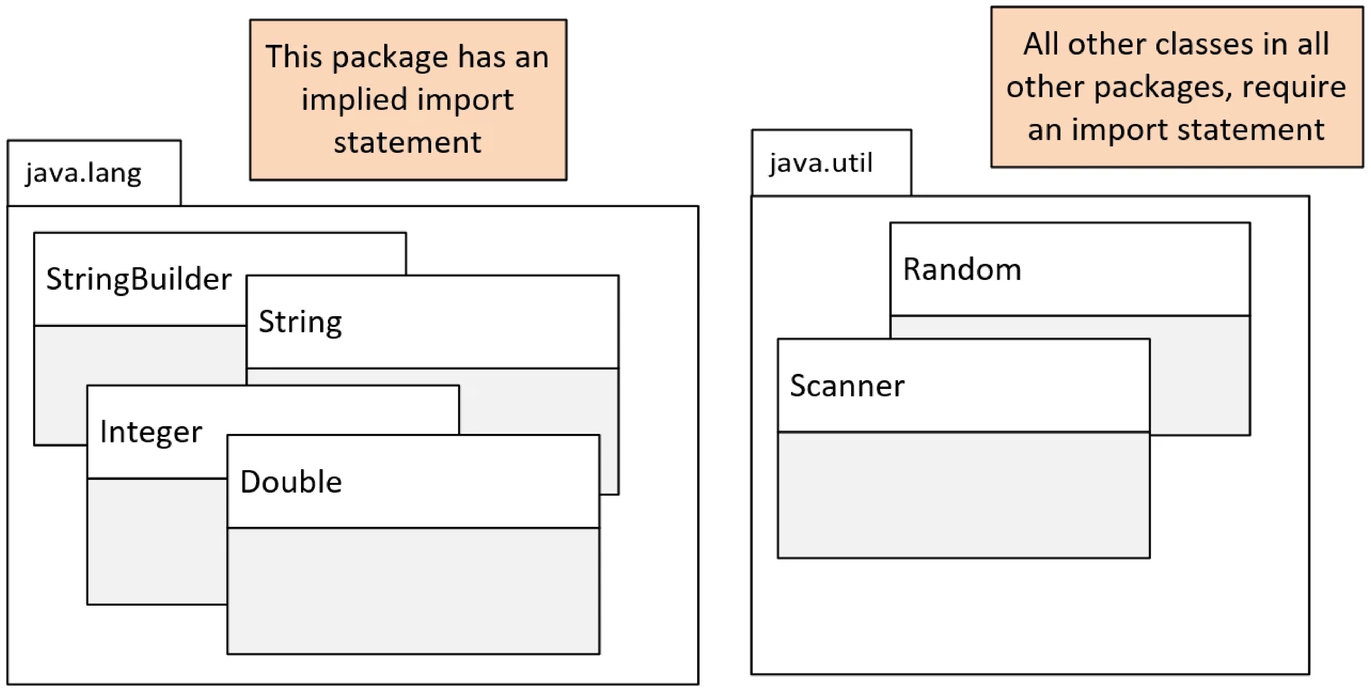
You can use any legal Java identifier for your package names, but common practice has package names as all lower case.

The period separates the hierarchical level of the package.

Java packages

By now, you're familiar with two of Java's packages, java.lang, and java.util.

On this slide, we show these two packages, with some of the classes we've used to date, or will use shortly, shown within these packages.



Using classes from packages other than java.lang - the import statement

You may remember, when we used the Scanner class or the Random class, we were required to use an import statement.

We show you an example of using the Scanner class in this code.

The import statement has to be declared before any class or type declarations, but after any package statement.

In this code, we don't have a package statement, so the import statement must be the first statement in the code.

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
    }
}
```

Multiple import statements

There is no limit to the number of import statements you can have.

This code shows two import statements, one for the Random class, and one for the Scanner class.

```
import java.util.Random;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
    }
}
```

Using import statements with wildcards

Alternately, we could use a wildcard, with the asterisks character, with the import statement.

In the following code, we're telling Java to import all classes from the java.util package, with the use of the asterisks, after the java.util package reference.

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
    }
}
```

What is the purpose of packages?

Packages let us re-use common class names across different libraries or applications, and provide a way to identify the correct class, either with an import statement, or a qualifying name.

For example, you might have a package for utility classes, that can provide common functionality, for all of your classes to access.

Packages let us organize our classes by functionality, or relationships.

Packages also let us encapsulate our classes, from classes in other packages.

So you might have a package of tightly coupled classes, that should only be accessed by each other, but not by the outside world, as an example.

What would a package name look like?

We've seen that Java starts their package names with java, in some of the examples we've looked at.

However, it is common practice, to use the reverse domain name to start your own package naming conventions.

If your company is abccompany.com for example, your package prefixes would be com.abccompany.

For the rest of the course, I'll be using dev.lpa, which is the reverse domain, of my Learn Programming Academy development company.

The package name hierarchy is separated by periods.

using the package statement

The package statement needs to be the first statement in the code, with the exception of comments.

The package statement comes before any import statements.

There can be only one package statement, because a class or type can only be in a single package.

```
package dev.lpa.package_one;

import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
    }
}
```

The Fully Qualified Class Name (FQCN)

In this code, we created a class called Main.

A class's fully qualified class name (FQCN) consists of the package name and the class name.

So this class's fully qualified name is, dev.lpa.package_one.Main.

```
package dev.lpa.package_one;

import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
    }
}
```

The Fully Qualified Class Name (FQCN)

It's unlikely this class, with it's fully qualified name, will have a naming conflict, with a Main class in another package.

As another example, the fully qualified class name of the Scanner class in this code, is java.util.Scanner.

```
package dev.lpa.package_one;

import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();
    }
}
```

The Fully Qualified Class Name vs the import statement

You can use the fully qualified class name, instead of the import statement as we show you here, on this slide.

This code does not use the import statement, but instead, where the Scanner class is referenced, the fully qualified class name is used.

```
package dev.lpa.package_one;

public class Main {

    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);

    }
}
```

The Fully Qualified Class Name vs the import statement

You can imagine this could get tedious, if you have to use the type often in your code.

Later in the course, we'll talk about using a combination of the import statement, and the fully qualified class name, to resolve conflicts.

```
package dev.lpa.package_one;

public class Main {

    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);

    }
}
```

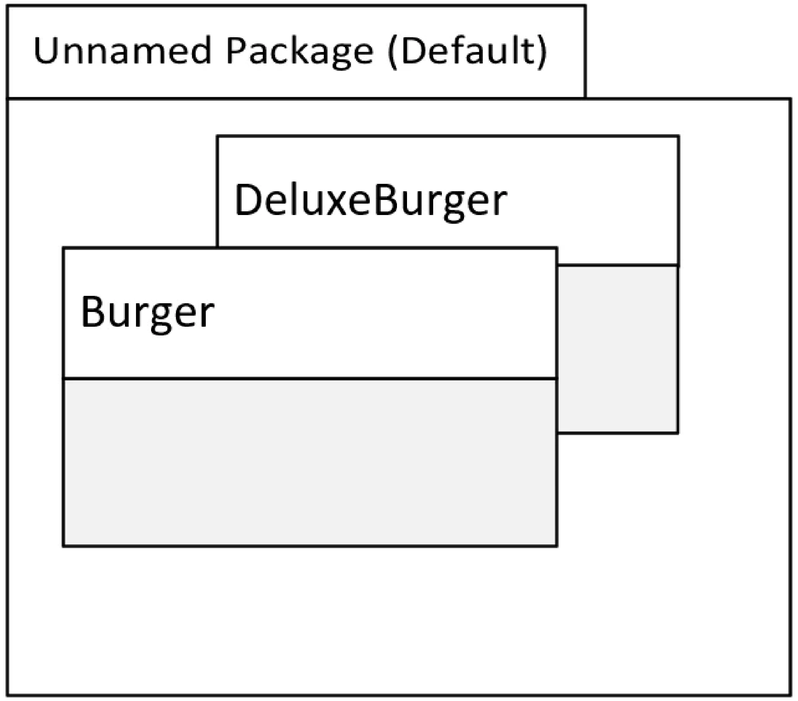
Using the package statement

In this slide, we're showing two classes from our previous challenge, in the unnamed or default package.

This is where classes are implicitly placed, if we don't specify a package statement.

For your applications, you should always specify a package statement, and avoid using the default or unnamed package.

Although that's all we've been using up until now, it has some disadvantages when you work in a true development environment.



Using the package statement

The main disadvantage is you can't import types from the default package into other classes, outside of the default package.

In other words, you can't qualify the name, if it's in the default package, and you can't import classes from the default package.