JAVA PROGRAMMING BASICS

Module 2: Java Object-oriented Programming

Training program

Module contents

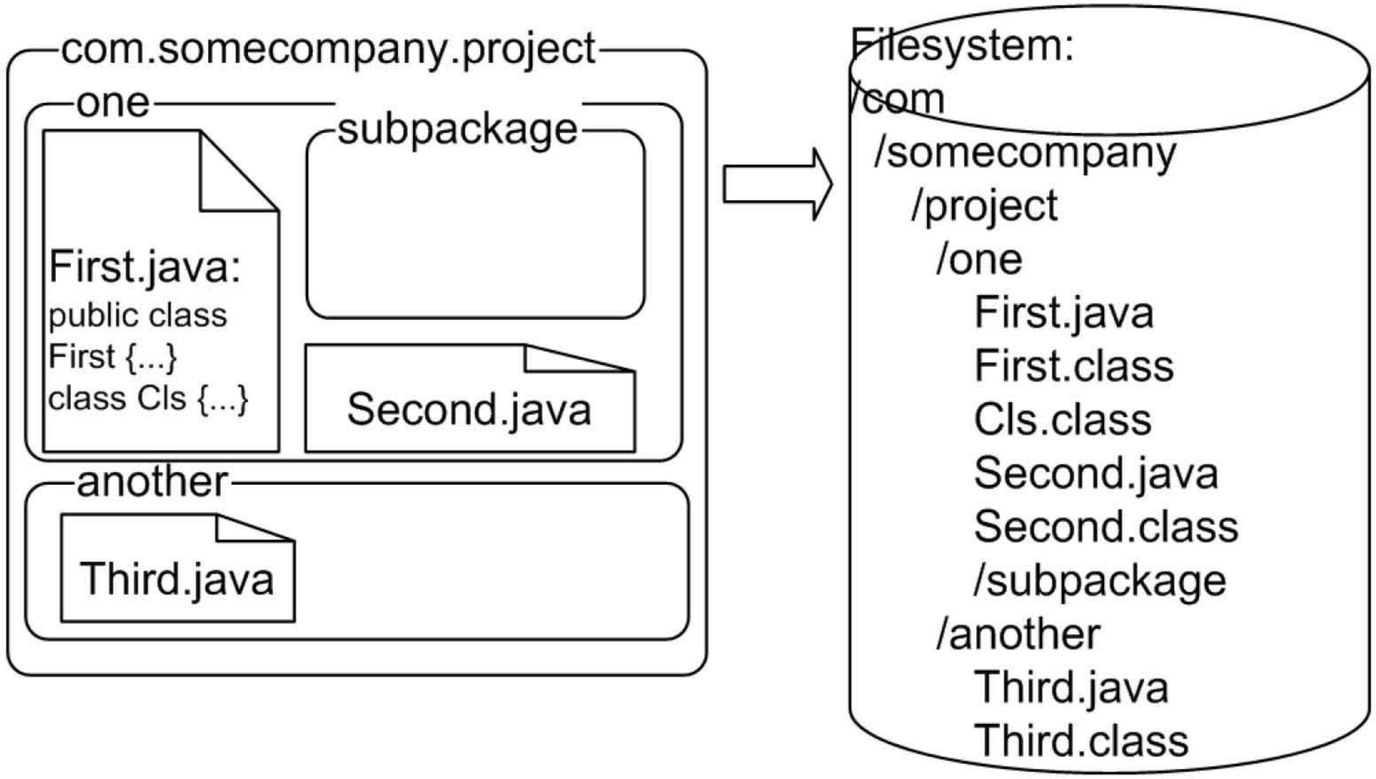
- Packages
  - The Package
  - Package import
  - Adding class to Package
  - Static import
  - Package organization
  - The jar utility
  - Executable jars

Module contents

- Packages
  - The Package
  - Package import
  - Adding class to Package
  - Static import
  - Package organization
  - The jar utility
  - Executable jars

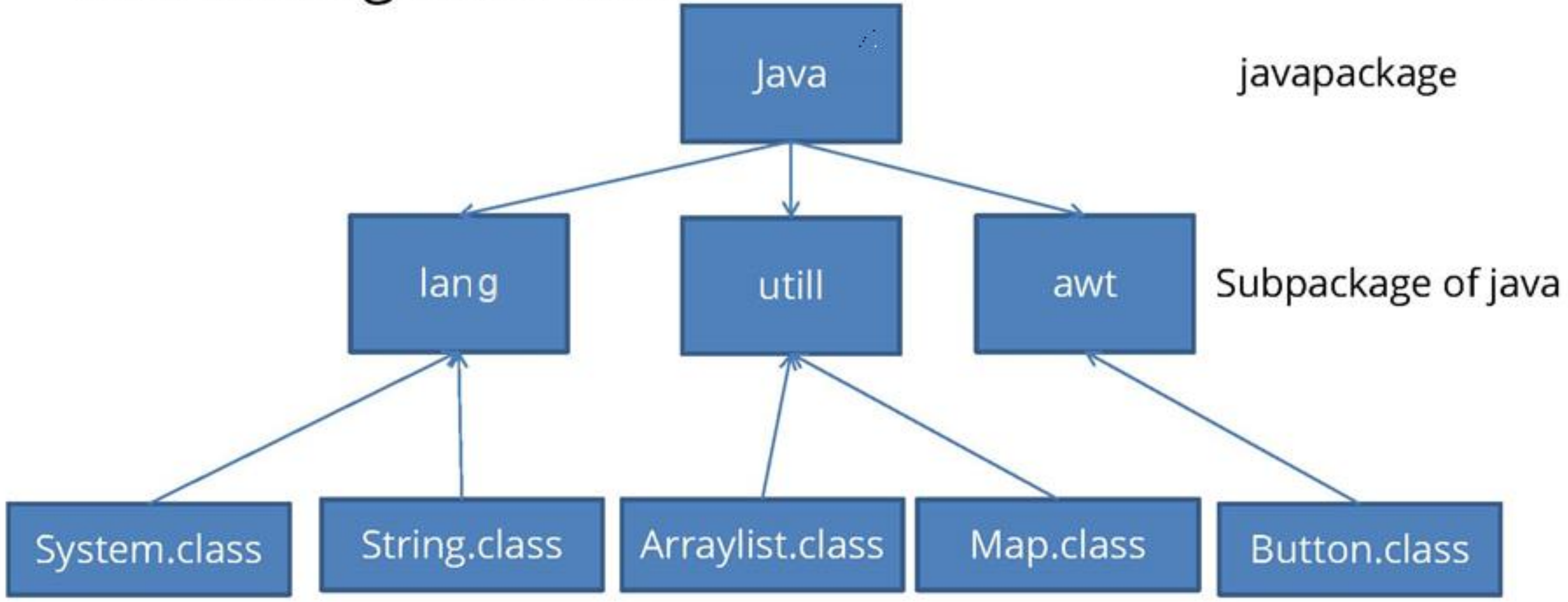
The Package 1/3

- A package is a collection of related classes and interfaces providing namespace management



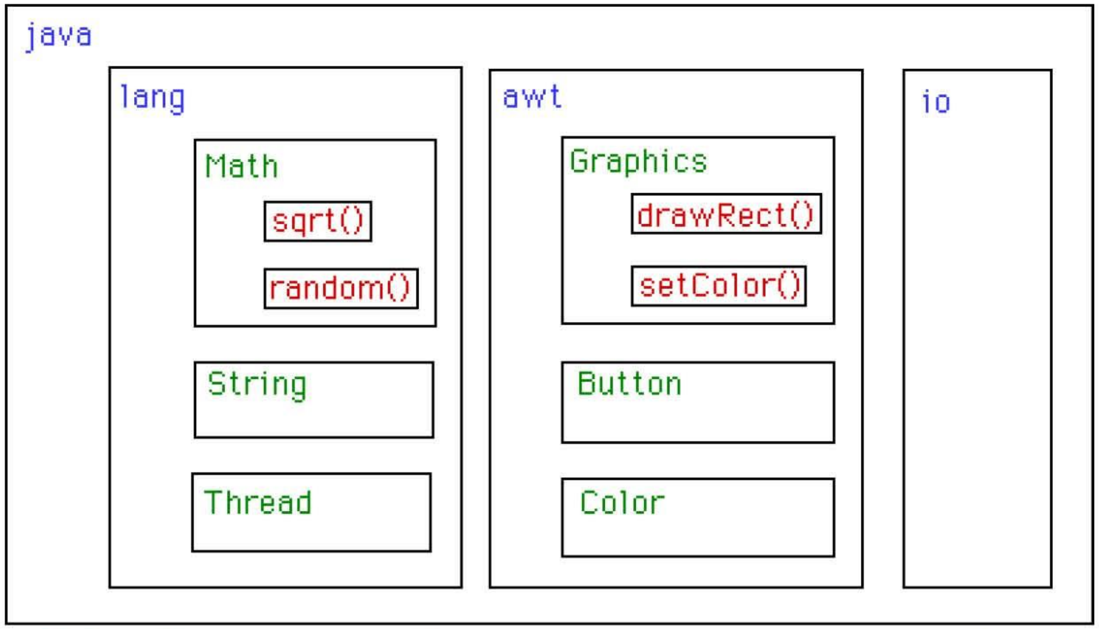
The Package 2/3

- Packages support hierarchical organization, and are used to organize large programs into logical and manageable units



The Package 3/3

- Java packages are namespaces. They allow programmers to create small private areas in which to declare classes.



Subroutines nested in classes nested in two layers of packages.

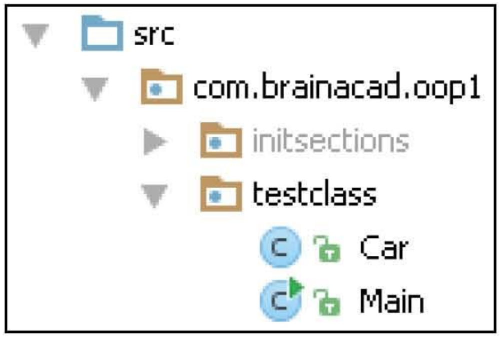
The full name of sqrt() is java.lang.Math.sqrt()

Module contents

- Packages
  - The Package
  - Package import
  - Adding class to Package
  - Static import
  - Package organization
  - The jar utility
  - Executable jars

Package import 1/4

- If class Main and Car are located in the same Java package:

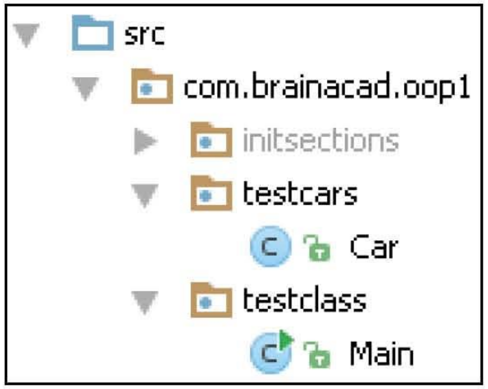


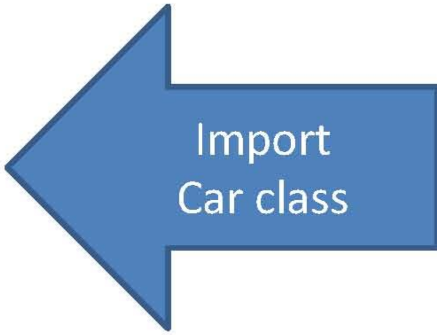
```
package com.brainacad.oop1.testclass;
public class Main {
    public static void main(String[] arg) {
        Car myCar1 = new Car();
    }
}
```

Package import 2/4

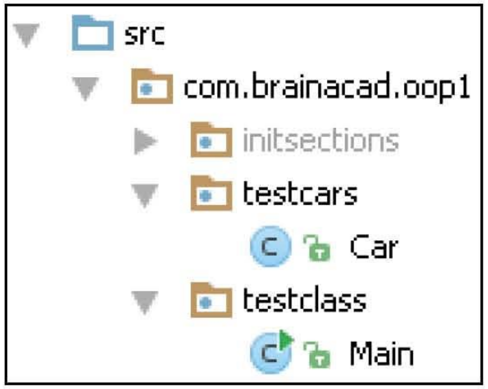
- If class Main and Car are located in the same Java package:

```
package com.brainacad.oop1.testclass;
import com.brainacad.oop1.testcars.Car;
public class Main {
    public static void main(String[] arg) {
        Car myCar1 = new Car();
    }
}
```



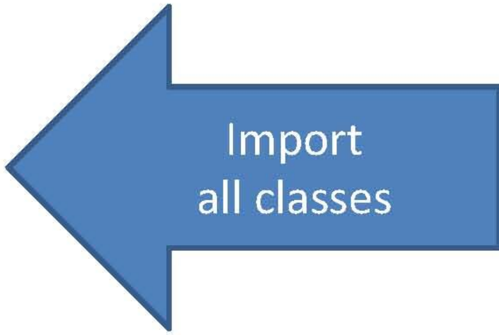


Package import 3/4

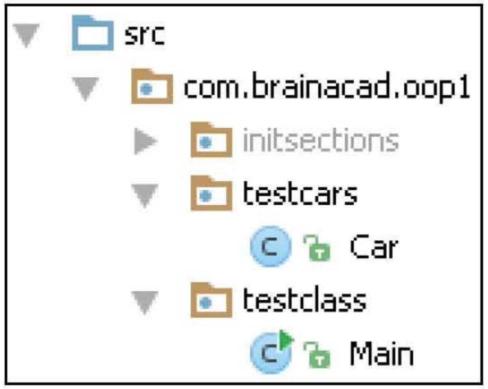


- If class Main and Car are located in the same Java package:

```
package com.brainacad.oop1.testclass;
import com.brainacad.oop1.testcars.*;
public class Main {
   public static void main(String[] arg) {
      Car myCar1 = new Car();
   }
}
```



Package import 4/4



- If class Main and Car are located in the different Java packages:

```java
package com.brainacad.oop1.testclass;
public class Main {
    public static void main(String[] arg) {
        com.brainacad.oop1.testcars.Car myCar1 =
            new com.brainacad.oop1.testcars.Car();
    }
}
```

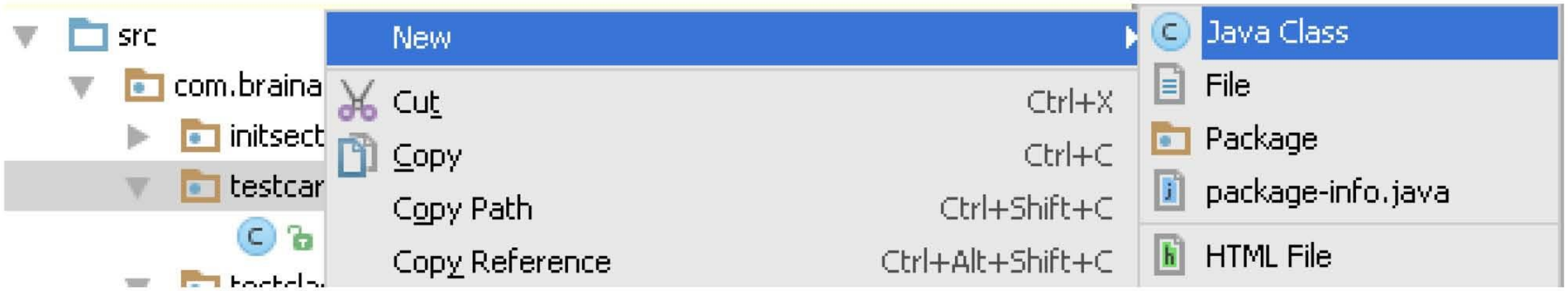
Fully Qualified Class Name

Module contents

- Packages
  - The Package
  - Package import
  - Adding class to Package
  - Static import
  - Package organization
  - The jar utility
  - Executable jars

Adding class to Package

- To create a new class in package



Module contents

- Packages
  - The Package
  - Package import
  - Adding class to Package
  - Static import
  - Package organization
  - The jar utility
  - Executable jars

Static import 1/3

```
package com.brainacad.oop1.testclass;
public class Main {
    public static void main(String[] arg) {
        double theta = 1;
        double r = Math.cos(Math.PI * theta);
    }
}
```

Static import 2/3

- Static imports allow the static items of one class to be referenced in another without qualification.

```
package com.brainacad.oop1.testclass;
import static java.lang.Math.PI;
import static java.lang.Math.cos;
public class Main {
    public static void main(String[] arg) {
        double theta = 1;
        double r = cos(PI * theta);
    }
}
```

Static import 3/3

- Static imports allow the static items of one class to be referenced in another without qualification.

```java
package com.brainacad.oop1.testclass;
import static java.lang.Math.*;
public class Main {
    public static void main(String[] arg) {
        double theta = 1;
        double r = cos(PI * theta);
    }
}
```

Module contents

- Packages
  - The Package
  - Package import
  - Adding class to Package
  - Static import
  - Package organization
  - The jar utility
  - Executable jars

Package organization 1/5

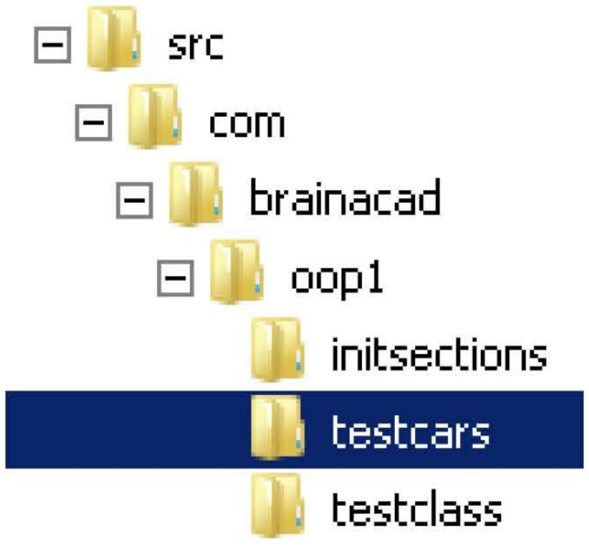
- **Package Naming Conventions**
- To prevent package name collisions, the convention is for organizations to use their reversed Internet domain names to begin their package names.
- For example, *com.brainacad*

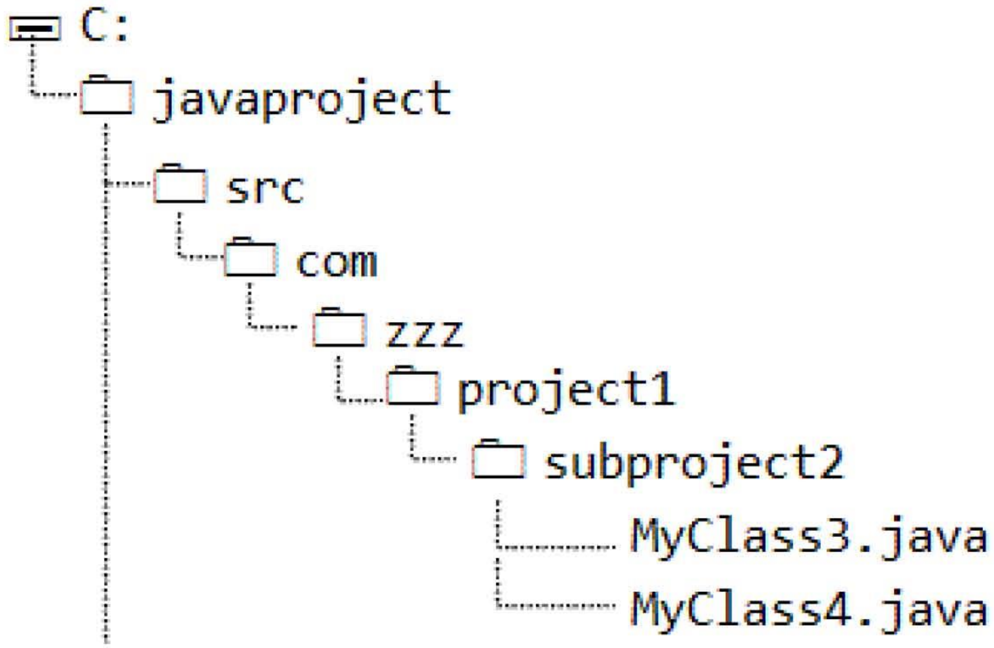
Package organization 2/5

- **Package Naming Conventions**
- If the Internet domain name contains an invalid character, such as a hyphen, the convention is to replace the invalid character with an underscore.
- If a domain name component starts with a digit or consists of a reserved Java keyword, the convention is to add an underscore to the component name.
- For example, *com.brainacad._1java*

Package organization 3/5

- The sub-directory structure corresponding to the package name for the classes will be created automatically if it does not already exist.





Package organization 4/5

- **Package By Feature**
- *Package-by-feature* uses packages to reflect the feature set. It tries to place all items related to a single feature (and *only*that feature) into a single directory/package. This results in packages with high cohesion and high modularity, and with minimal coupling between packages.
- *com.mycompany.report*
- *com. mycompany.security*
- *com. mycompany.util*

Package organization 5/5

- **Package By Layer**
- The competing *package-by-layer* style is different. In package-by-layer, the highest level packages reflect the various application "layers", instead of features, as in:
- *com. mycompany.action*
- *com.mycompany.model*
- *com. mycompany.dao*

Module contents

- Packages
  - The Package
  - Package import
  - Adding class to Package
  - Static import
  - Package organization
  - The jar utility
  - Executable jars

The jar utility

- jar-The Java Archive Tool

- jar combines multiple files into a single JAR archive file.

- Java Archive (JAR) is a platform-independent file format that allow you to compress and bundle multiple files associated with a Java application, into a single file. JAR is based on the popular ZIP algorithm, and mimic the Unix's tar file format

The jar utility

- Packaging project to Jar-file in IDE
- Run program from jar-file
- Explore Jar-file

Use TestCar project

- Connect jar-file as other project dependency in IDE
- Packaging project with dependency to Jar-file in IDE with "extract to target JAR" option
- Run program from jar-file
- Explore Jar-file
- Packaging project with dependency to Jar-file in IDE with "copy to the output directory and link via manifest" option
- Run program from jar-file
- Explore Jar-file

Use ImportFromJarApp project