5. PACKAGES

1. Package Definition :

The package is the container for classes and interfaces (.class files)
In simple, it is a way of categorizing the classes and interfaces. When developing applications in Java, hundreds of classes and interfaces will be written, therefore categorizing these classes is a must as well as makes life much easier.

2. Visibility Control :

Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are:

- **private :** Visible to the class only.
- **default (no access modifier) :** Visible to the package, the default.
- **public :** Visible to the world.
- **protected :** Visible to the package and all subclasses from outside

Access By	private	package	protected	public
the class itself	yes	yes	yes	yes
a subclass in same package	no	yes	yes	yes
non-subclass in same package	no	yes	yes	yes
a subclass in other package	no	no	yes	yes
non-subclass in other package	no	no	no	yes

3. Uses of Packages:

- **Grouping and simplicity :** the classes with similar functionality can be stored in same package. It provides simplicity to access the classes. e.g. In java all utility classes are stored in util package.
- **Security and Visibility control :** provides security to our own classes and interfaces. Means if we make them default then those can not be accessed outside of package. members with no access modifier are visible only within the package
- **Naming control**: to avoid name collision: Classes or interfaces may have same names in different packages.
- **Code reusabilty :** by importinmg the packages anywhere we can use the classes or interfaces defined in that package. So we can reuse the code anywhere.

4. Defining, finding and importing packages :

4.1. Defining a package:

5. PACKAGES

When we want to define our own packages following syntax should be used while writing the code.

1. Defining simple package: To include the class in a package we need to write the package statement

Syntax:

```
package package-name;
```

Example:

```
package mypack;
```

2. Defining hierarchical package:

Syntax:

```
package outerpackagename.innerpackagename;
```

Example:

```
package mypack.mymath;
```

Note: package statement must be first statement when u are wrting the class under the package. If we do not create our own package then java provides default package with no name. The classes defined by package, should be stored in the same named container as package name.

Example:

```
package mypack;
class PackDemo
{
    // data and code
}
```

The .class file of above class should be stored into folder mypack.

4.2. Finding a package:

To import the package it should be searchable.
Bydefault, packages are searchable into our own working directory. If the package is anywhere else we have to set classpath upto the previous directory where packages are stored.

For example, if above defined package mypack is in the following directory

```
c://CoreJava/mypack
```

then the class path should be set up to the CoreJava. The command to set class path on windows is:

```
set classpath=c://CoreJava
```

4.3. Importing a package:

Once the package is searchable, we can import the package by using following syntax.

5. PACKAGES

1. Importing a single package:

Syntax:

```
import packagename.classname;
```

Example:

```
import mypack.PackDemo;
```

2. Importing Hierarchical package:

Syntax:

```
import  outerpackagename.innerpackagename;
```

Example:

```
import mypack.mymath.PackDemo;
```

If we want to import all the classes and interfaces from a package then we can use following syntax:

Syntax:

```
import  packagename.*;
import  outerpackagename.innerpackagename.*;
```

Example:

```
import  mypack.*;
import mypack.mymath.*;
```

Example on packages:

Vehical.java

```
package mypack;              //defining userdefined package
class Vehical
{
        public void getVehicalInformation() {
                // code
        }
        public void setVehicalInformation() {
                //code
        }
}
```

Car.java

```
import java.io.IOException;      // importing inbuilt package
import mypack.Vehical;        // importing user defined package
abstract class Car exntends Vehical {
        static int i=9;
        int k=8; String carname;
```

```
        Car() {
                System.out.println("In abstract class constructor");
        }

        public void setData(String carname)throws IOException {
                this.carname=carname;
        }
}
```

5. Static Import:

A new feature to Java called static import expands the capabilities of the import keyword. The static import statement can be used to import the static members of a class or interface. When using static import, it is possible to refer to static members directly by their names, without having to qualify them with the name of their class. This simplifies and shortens the syntax required to use a static member.

Syntax:

```
import static java.lang.Math.sqrt;
import static java.lang.Math.pow;
```

The methods sqrt and pow are static methods of class Math. By static import we can directly access these methods without using classname.

Example:

without static import we have to call these methods like:

```
        Math.sqrt(9);
        Math.pow(2,4);
```

but using static import we can directly access like:

```
        sqrt(9);
        pow(2,4);
```

Thus packages are important aspect in java to store your own .class files.

ASSIGNMENTS

1. Store the Vehical class into package named user, Car class in package named type, Duster class in package named model.
2. Control the visibility of some of the variables wihtin the package only.
3. Import appropriate classes and show the execution of above inheritance. Also show the use of static import.