Need of Packages

The main feature of Object Oriented Programming is the ability to reuse the code. There few ways of achieving this by Inheritance, Interface but this is limited to reusing the classes within a program.

If we want to use classes from the other programs without physically copying the classes into the program, then this can be only achieved by the use of package.

Introduction

- Java package is a way of grouping a variety of classes, interfaces and sub-packages.
- The grouping is usually done according to functionality.

Advantages of Package

1. **Reusability of code**

   Reusability saves time, effort and also ensures consistency. A class once developed can be reused by any number of programs.

2. **Bundle classes and interfaces**

   Package is used to categorize the classes and interfaces so that they can be easily maintained.

3. **Classes Isolation**

   Classes in packages can have their own data members and member functions that are visible by all classes inside the package, but not outside.

4. **Access Protection**

   Packages can contain hidden classes that are used by the package but are not visible or accessible outside the package.

5. **Removing Naming Collision**

   Different packages can have classes with the same name. Example: java.awt.Frame and photo.Frame.

6. **Smaller Size**

   Java packages can be stored in compressed files called JAR files.

Disadvantages of Package

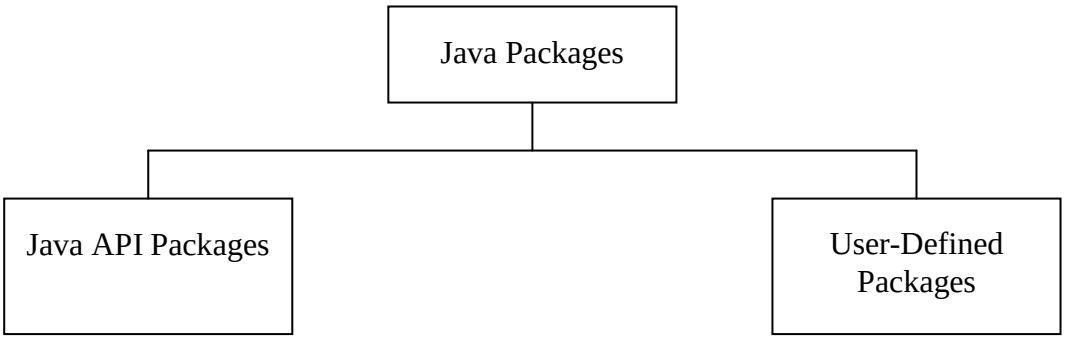
1. **Passing parameters**

   We cannot pass parameters to packages.

2. **Overhead in changing function definition**

   Change in a function needs to be reflected in all the functions that use the changed function and hence the whole package needs to be recompiled.

Types of Packages



1. **Java API Packages**

   They are also known as Built-in packages. They are already defined package like java.io.*; java.lang.*; etc.

2. **User-Defined packages**

   These packages are created by user itself.

Creating Package

Syntax:

package <package_name>;

Note: Class files must be saved in separate sub-folder with the same name as <package_name>.

Accessing Package

There are three ways to access the package:

1. import package.classname;
2. import package.*;
3. fully qualified name

1. import package.classname;

- Only declared class of the package will be accessible.

Example:

first.java

package pack;
public class first

```
{
	public void displayA()
	{
		System.out.println("I am in first class");
	}
}
```

second.java

```
package pack;
public class second
{
	public void displayB()
	{
		System.out.println("I am in Second class");
	}
}
```

check.java

```
import pack.first;
import pack.second;
class check
{
	public static void main(String args[])
	{
		first obj1=new first();
		obj1.displayA();
		second obj2=new second();
		obj2.displayB();
	}
}
```

Output

```
I am in first class
I am in second class
```

2. import package.*;

- All the classes and interfaces of this package will be accessible but not sub-packages.

Example:

first.java

```
package pack;
public class first
{
	public void displayA()
	{
		System.out.println("I am in first class");
	}
}
```

second.java

```
package pack;
public class second
{
	public void displayB()
```

```
	{
		System.out.println("I am in Second class");
	}
}
```

check.java

```
import pack.*;
class check
{
	public static void main(String args[])
	{
		first obj1=new first();
		obj1.displayA();
		second obj2=new second();
		obj2.displayB();
	}
}
```

Output

```
I am in first class
I am in second class
```

3. Fully Qualified Name

- No need to import package
- While accessing the class or interface, every time we need to use fully qualified name.
- Only declared class of the package will be accessible.

Example:

first.java

```
package pack;
public class first
{
	public void displayA()
	{
		System.out.println("I am in first class");
	}
}
```

second.java

```
package pack;
public class second
{
	public void displayB()
	{
		System.out.println("I am in Second class");
	}
}
```

check.java

```
class check
{
	public static void main(String args[])
	{
```

```
		pack.first obj1=new pack.first();
		obj1.displayA();
		pack.second obj2=new pack.second();
		obj2.displayB();
	}
}
```

Output

```
I am in first class
I am in second class
```

Static Import

- It facilitate programmer to access any static member of a class directly.
- There is no need to qualify it by the class name.

Advantage of static import:

- Less coding.

Disadvantage of static import:

- Overuse of static import makes the program unreadable and un-maintainable.

Example:

first.java

```
package pack;
public class first
{
	public static void displayA()
	{
		System.out.println("I am in first class");
	}
}
```

second.java

```
package pack;
public class second
{
	public static void displayB()
	{
		System.out.println("I am in Second class");
	}
}
```

check.java

```
import static pack.first.*;
import static pack.second.*;
class check
{
	public static void main(String args[])
	{
		displayA();
		displayB();
```

```
        }
}
```

Output

```
I am in first class
I am in second class
```

Questions asked in semester paper

Question-What are Packages in java? How a user defined package is created in java, explain with example?

[2017-2018]

Question-How do you call a package?

[2016-2017]