OBJECT ORIENTED PROGRAMMING USING JAVA

OUTLINE

- PACKAGES IN JAVA

PACKAGES IN JAVA

- A java package is a group of similar types of classes, interfaces and sub-packages.
- Think of it as a folder in a file directory.
- Package in java can be categorized in two form, built-in package and user-defined package.
- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Banking Application

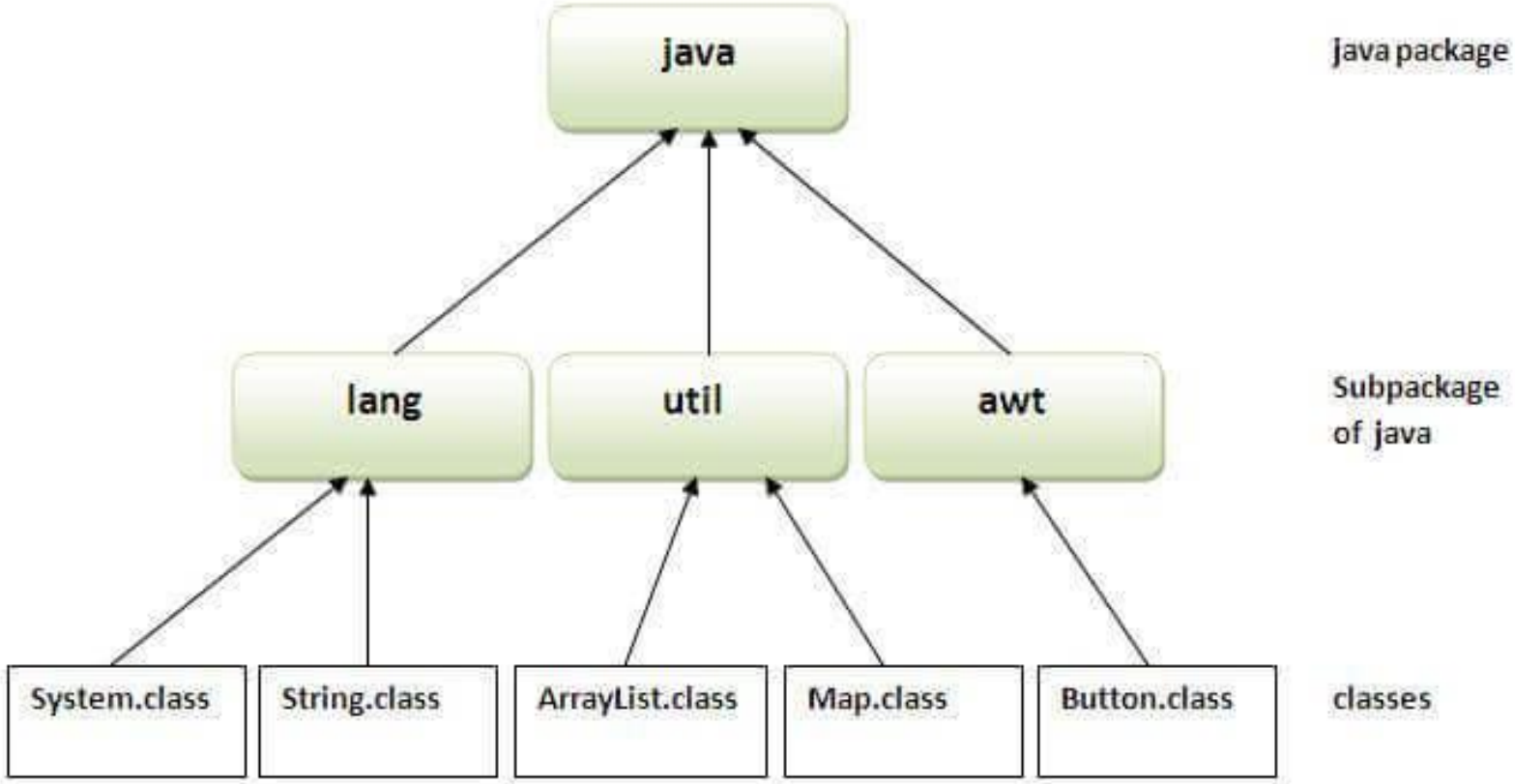
ACCOUNTS (PACKAGE) *TRANSACTION(PACKAGE)* *LOAN(PACKAGE)*
.CLASS FILE *.CLASS FILE* *.CLASS FILE*

.CLASS FILE *.CLASS FILE* *.CLASS FILE*

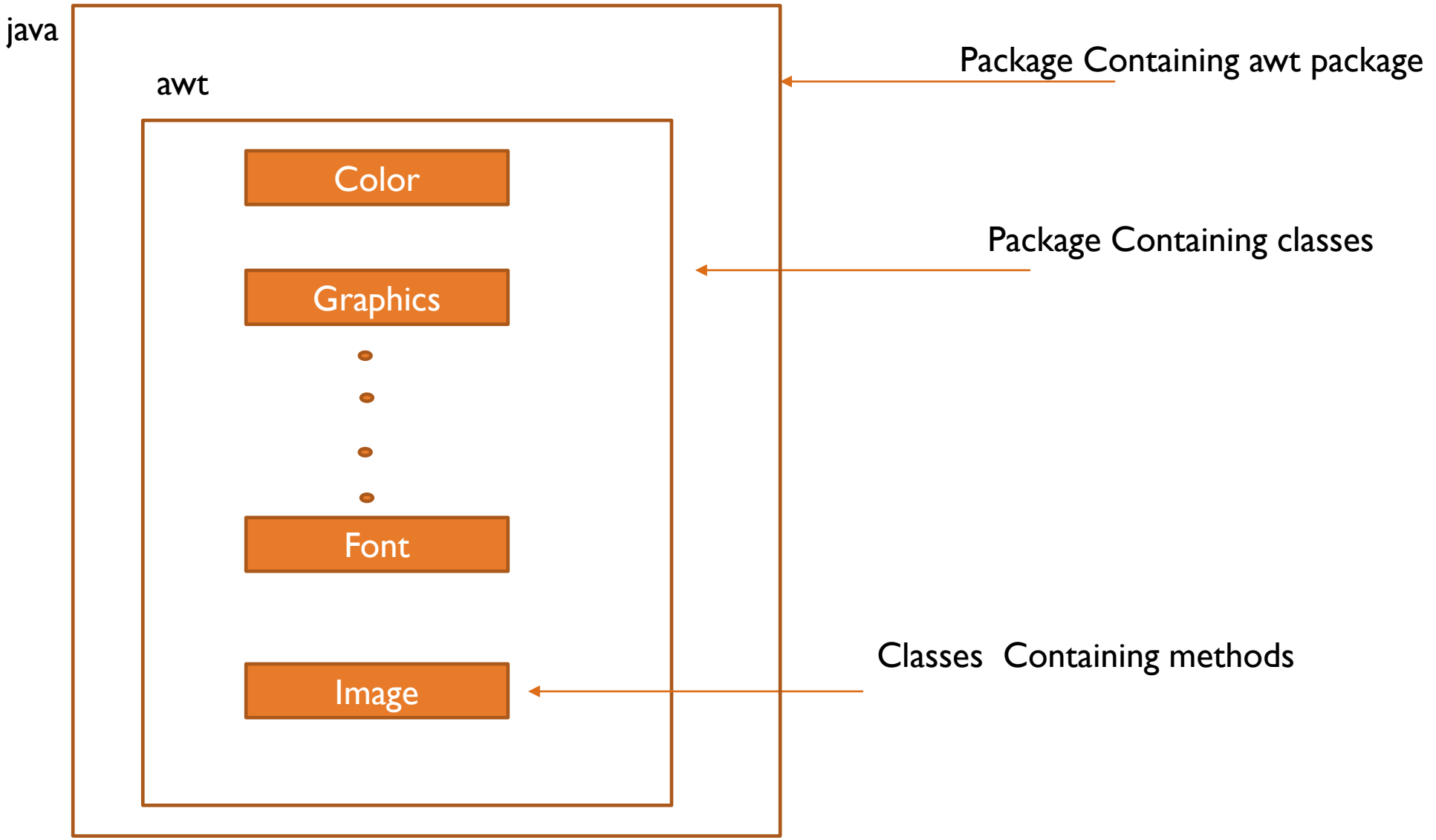
ADVANTAGE OF JAVA PACKAGE

- The classes contained in the packages of the other programs can be easily reused.
- Java package is used to categorize the classes and interfaces so that they can be easily maintained.
- Java package provides access protection.
- Java package removes naming collision.

PACKAGE : EXAMPLE



HIERARCHICAL REPRESENTATION OF JAVA.AWT PACKAGE



CREATING PACKAGE: EXAMPLE

```
//save as Simple.java
package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Welcome to package");
   }
}
```

- **NOTE: If no package is specified, the classes in the file goes into a special unnamed package (the same unnamed package for all files).**

HOW DOES THE JAVA RUN-TIME SYSTEM KNOW WHERE TO LOOK FOR PACKAGES THAT YOU CREATE?

- First, by default, the Java run-time system uses the current working directory as its starting point. Thus, if your package is in a subdirectory of the current directory, it will be found.
- Second, you can specify a directory path or paths by setting the CLASSPATH environmental variable.
- Third, you can use the-classpath option with java and javac to specify the path to your classes.

PREDICT THE OUTPUT?

```
package pack1;
package pack2;
public class Main
{
}
```

PREDICT THE OUTPUT?

```
package pack1;
package pack2;
public class Main
{
}
```

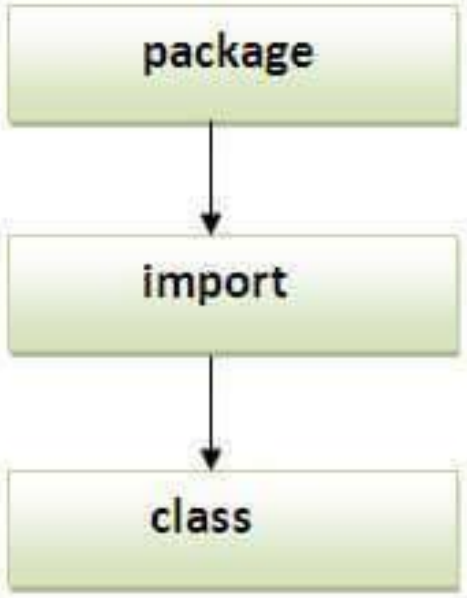
```
Main.java:10: error: class, interface, or enum expected
package pack2;
^
1 error
```

Note: Only one package is valid in java, more than one package in a single source code invalid in java

NAMING CONVENTIONS

- Package can be named using standard java naming rule. Where it begins with lowercase letters. This makes easy to distinguish between package names and class name.
- **Example:**
- double y= **java.lang.Math.sqrt();**

SEQUENCE OF THE PROGRAM:



- Note: If you import a package, all the classes and interface of that package will be imported excluding the classes and interfaces of the subpackages. Hence, you need to import the subpackage as well.

PREDICT THE OUTPUT?

```
import java.util.*;
package pack1;
class Main
{
}
```

PREDICT THE OUTPUT?

```java
import java.util.*;
package pack1;
class Main
{
}
```

```
Main.java:9: error: class, interface, or enum expected
package pack1;
^
1 error
```

HOW TO ACCESS PACKAGE FROM ANOTHER PACKAGE?

There are three ways to access the package from outside the package.

Implicit class import:- import package.*;
Explicit class import:- import package.classname; **(improves readability of code)**
Fully qualified name.

Note: after package name use .*
after class name use ;

WHICH IS CORRECT?

1. **import java.util.Scanner;**
2. **import java.util. Scanner.*;**
3. **import java.util.*;**
4. **import java.util;**

WHICH IS CORRECT?

1. **import java.util. Scanner;(Correct)**
2. import java.util. Scanner.*;(Wrong)
3. **import java.util.*;(correct)**
4. import java.util;(wrong)

I) USING PACKAGENAME.*

1. **If you use package.* then all the classes and interfaces of this package will be accessible but not sub-packages.**
2. **The import keyword is used to make the classes and interface of another package accessible to the current package.**

EXAMPLE

```
//save by A.java
package pack;
public class A{
  public void msg()
          {System.out.println("Hello");}
}
//save by B.java
package mypack;
import pack.*;
class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

Output:
HELLO

2) USING PACKAGENAME.CLASSNAME

If you import package.classname then only declared class of this package will be accessible.

//save by A.java

```
package pack;
public class A{
  public void msg(){System.out.println("Hello");
}
}
```

Output:Hello

//save by B.java

```
package mypack;
import pack.A;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

3) USING FULLY QUALIFIED NAME

- If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.

- It is generally used when two packages have same class name e.g. java.util and java.sql packages contain Date class.

EXAMPLE:

//save by A.java

```
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
```

Output:Hello

//save by B.java

```
package mypack;
class B{
  public static void main(String args[]){
   pack.A obj = new pack.A();//using fully qualified name
   obj.msg();
  }
}
```

EXAMPLE:

```
class test
{
Public static void main(string[] args)
{
Java.util.Arraylist l=new
java.util.ArrayList();
}
}
```

Note:

- **Problem with readability**
- **Increase length code**
- **Solve this problem using import statement. It is no required to use fully qualified name we can use short name directly.**

EXAMPLE:

```
import java.util.*;
import java.sql.*;
class Main
{
    public static void main(String args[])
    {
        Date d=new Date();
    }
}
```

EXAMPLE:

```
import java.util.*;
import java.sql.*;
class Main
{
    public static void main(String args[])
    {
        Date d=new Date();
    }
}
```

```
Main.java:14: error: reference to Date is ambiguous
        Date d=new Date();
        ^
  both class java.sql.Date in java.sql and class java.util.Date in java.util match
```

MULTIPLE PACKAGE SCENARIO:

- In that scenario:
- Highest priority will go to explicit class import
- Next level will goes to classes present in current working directory (default package)
- Implicit class import

EXAMPLE:

```
import java.util.Date;
import java.sql.*;
class Main
{
    public static void main(String args[])
    {
        Date d=new Date();
    }
}
```

Gives priority to explicit import

This code executes fine

WHICH ONE IS VALID?? (IMPORTING A PACKAGE TO USE PATTERN CLASS IN PROGRAM)

- **Java-> util-> regrex-> pattern**

1. **Import java.*;**
2. **Import java.util.*;**
3. **Import java.util.regex.*;**
4. **No import required**

WHICH ONE IS VALID?? (IMPORTING A PACKAGE TO USE PATTERN CLASS IN PROGRAM)

- **Java-> util-> regrex-> pattern**

1. **Import java.*;**
2. **Import java.util.*;**
3. **Import java.util.regex.*;**
4. **No import required**

DEFAULT: PACKAGES AVAILABLE IN JAVA

- Java.lang package is by default package available in all java program hence we don’t need to write import.

```
class example
{
public static void main (String[] Args)
String s= new String(“main”);
}
}
```

STATIC IMPORT

- According to **sun**: usage of static import reduces code improves readability
- But according to **www expert** (like us): usage of static import creates confusion and reduces readability
- Hence if there is not specific requirement it is not recommended to use static import.
- for static import:
- Explicit (import static packagename.classname.staticmember

  example import static java.lang.math.sqrt;
- Implicit (import static package name .classname.*)

  example:import static java.lang.math.*;

EXAMPLE

Without static import

```
class Main
{
public static void main (String[] Args)
{

System.out.println(Math.sqrt(4));
System.out.println(Math.max(10,20));
System.out.println(Math.random());

}
}
```

With static import

```
import static java.lang.Math.*;
class Main
{
public static void main (String[] Args)
{

System.out.println(sqrt(4));
System.out.println(max(10,20));
System.out.println(random());

}
}
```

```
2.0
20
0.6706409964209594
```

EXAMPLE

With static import

```
import static java.lang.Math.sqrt;
class Main
{
public static void main (String[] Args)
{

System.out.println(sqrt(4));
System.out.println(max(10,20));
System.out.println(random());

}
}
```

```
Main.java:15: error: cannot find symbol
System.out.println(max(10,20));
                   ^
  symbol:   method max(int,int)
  location: class Main
Main.java:16: error: cannot find symbol
System.out.println(random());
```

EXAMPLE

Class system (present in java lang)
{
Static printstream out; // (out is a static variable present in system class of the type print stream)
}

System.out.println() //(print is method present in the system class)

EXAMPLE

```
import  static java.lang.System.out;
class Main{
public static void main (String[]  args)
{
 out.println("hello");
out.println("hi");
}
}
```

```
hello
hi
```

WHICH PACKAGE IS VALID?

- import java.lang.Math.*;
- import static java.lang.Math.*;
- import java.lang.Math.sqrt*;
- import static java.lang.Math.sqrt();
- import java.lang.Math.sqrt;
- import static java.lang.Math.*;
- import java.lang;
- import static java.lang;
- import java.lang.*;
- import static java.lang.*;

WHICH PACKAGE IS VALID?

- import java.lang.Math.*; (invalid after class ; not star)
- import static java.lang.Math.*;(valid)
- import java.lang.Math.sqrt*; (invalid : will take, till class level not till data member)
- import static java.lang.Math.sqrt(); (invalid not bracket open or close)
- import java.lang.Math.sqrt; (invalid)
- import static java.lang.Math.*;**(valid)**
- import java.lang; (invalid .* required)
- import static java.lang; (invalid always talks about class and data member)
- **import java.lang.*; (valid)**
- import static java.lang.*;(invalid)

THANK YOU
?