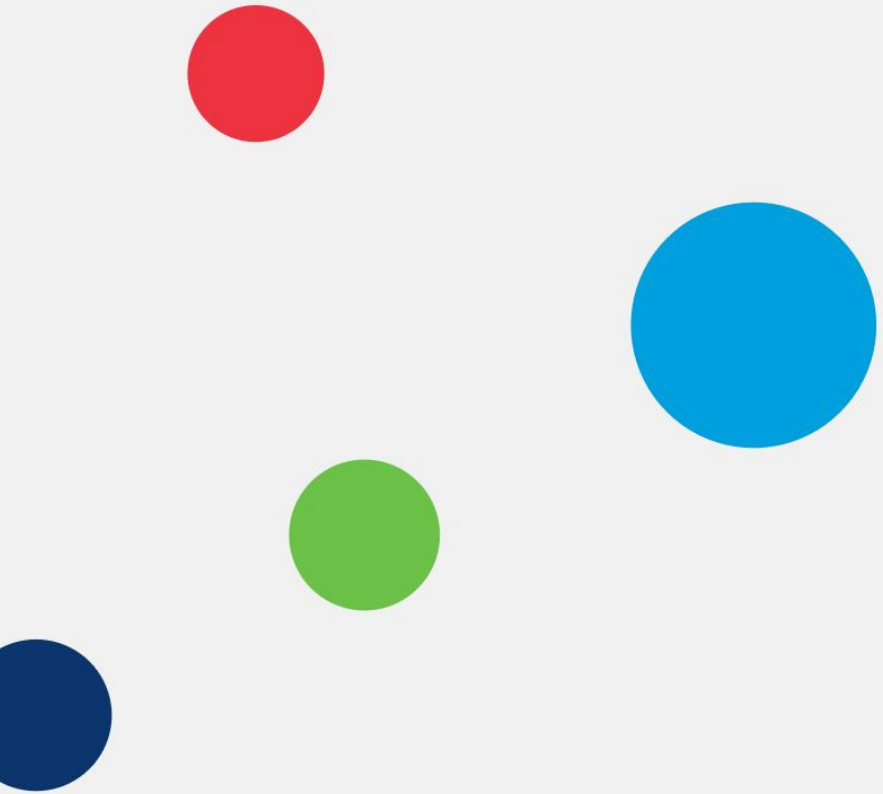

# Import, Static Import & Creating our own Packages

# Agenda

1 **Static Import**

# Importing Classes



# Packages & import statement

- Java provides import statement.
  - Import means, we are including the classes and interfaces of existing packages in our program.
  - If you need a sub package, then, you need to issue a separate import statement.

- For example,
  - `import java.awt.*;` -- this will be importing awt package
  - `import java.awt.event.*;`
    - this will be importing event package which is a sub package under awt package.

# Importing Classes from Packages

- Java has used the package mechanism extensively to organize classes with similar functionality in one package
- If you want to use these classes in your applications, you can do so by including the following statement at the beginning of your program:
  - `import packagename.classname;`
- *If the packages are nested you should specify the hierarchy.*
  - `import package1.package2.classname;`

# Importing Classes from Packages (Contd.).

- The class you want to use must be qualified by its package name.
- If you want to use several classes from a package, it would be cumbersome to type so many classes qualified by their packages.
- It can be made easy by giving a star(*) at the end of the import statement. For example:

```
import  package1.*;
```

# Static Import

- A static import declaration enables us to refer to imported static members as though they were declared in the current class

- If we use static import, we first have to import this static member in the following way :

```
package p1;
public class Abc {
     public static void xyz() {
     System.out.println("static import demo");
     }
}
```

```
package p2;
import static p1.Abc.xyz;
public class A1 {
     public static void main(String[] args) {
          xyz();
     }
}
```

Output : "static import demo"

# Static Import (Contd.).

- If we are invoking multiple static members of the same class, we can also use asterisk(*), which indicates that *all* static members of the specified class should be available for use

```
import static java.lang.Math.*;
public class StaticImportDemo  {
     static float x = 4.556f;
     static double y =4.556D;
     public static void main( String args[] )   {
       float a1 = abs(x);
       int r1  = round(x);
       double s1 = sqrt(y);
       System.out.println("absolute value of "+x+" is" +a1);
       System.out.println("When we round off "+x+"we get" +r1);
       System.out.println("Square Root of "+y+ "is" +s1);
     }
   }
```



# Quiz

Which is the correct usage of import statement?

A) import java.*;

B) import java.lang.*;

C) import *;

D) import *.*;

Which Option is correct ?



# Understanding CLASSPATH

What is CLASSPATH?

- CLASSPATH is an environment variable that tells the Java runtime system where the classes are present
- The current directory is the default directory for CLASSPATH.

# Understanding CLASSPATH (Contd.).

- When you create your own package for example MyPack, all the .class files are saved in the directory MyPack.

- In order for a program to find MyPack, one of two things must be true:
  - Either the program is executed from a directory immediately above MyPack,
  - or
  - CLASSPATH must be set to include the path to MyPack folder

# Creating our own Package Example

```
package empPack;
class Emp {
  String empName;
  double salary;

  Emp (String name, double sal){
    empName = name;
    salary = sal;
  }
  void display(){
    System.out.println(empName + " : $"+salary);
  }
}
```

# Quiz

In one java source file, how many package statements can be used?

A) One

B) Two or more

Only Option A is correct;
You can't have two or more package statements in a java source file



# Creating our own Packages

We can create our own packages in java

- Package statement helps us to create our own package.
- Package statement should be the first statement in your program.
- We group related classes and interfaces into a package
- We can have sub-packages inside our packages as required

Packages are stored as directories in Hard disk:

- Remember, the case should match exactly
- Look at the program in next page & try it from command line:

# Working with Packages – Example 1

```
package automobile;


public class Vehicle {
public void printname() {
        System.out.println("My name is vehicle");
        System.out.println(" I am defined inside automobile
  package");
    }
}
```

What is the package name?
How you will save this file?

# Working with Packages – Example 1 (Contd.).

```
package automobile;

public class Bike extends Vehicle {
    public void printname() {
    System.out.println("My name is bike");
    System.out.println(" I am defined inside automobile
  package");
    }
}
```

What is the package name?
How you will save this file?

## Working with Packages – Example 1 (Contd.).

```
package automobile;

public class Car extends Vehicle {
    public void printname() {
        System.out.println("My name is car");
        System.out.println(" I am defined inside
  automobile package");
    }
}
```

What is the package name?
How you will save this file?

# Working with Packages – Example 1 (Contd.).

```
package au_test;
import automobile.*;
public class tester {
public static void main(String s[ ] ) {
System.out.println(" I am tester class defined inside au_tester
  package");
System.out.println(" I had imported all classes of automobile
  package");
System.out.println(" Creating instances of Vehicle, Car and Bike ");
```

# Working with Packages – Example 1 (Contd.).

```
Vehicle v = new Vehicle();
Car c = new Car();
Bike b = new Bike();
System.out.println(" Accessing the functions using objects");
 v.printname();
 c.printname();
 b.printname();
}
}
```

How you will save this file?
Then, In command prompt:
How you will compile?
And How you will run?

What is the output of the program?

# What will be the result, when you try to compile and execute :

```
class A1 {
    protected void m1() {
        System.out.println("m1 method of class A1");
    }
}
class A2 extends A1 {
    void m1() {
        System.out.println("m1 method of class A2");
    }
    public static void main(String[] args) {
       A2 x = new A2();
       x.m1();
    }
}
```

Compilation Error…Why?

# What will be the result, when you try to compile and execute (Contd.).

```
class A1 {
     protected void m1() {
          System.out.println("m1 method of class A1");
     }
}
class A2 extends A1 {
     public void m1() {
          System.out.println("m1 method of class A2");
     }
     public static void main(String[] args) {
        A2 x = new A2();
        x.m1();
     }
}
```

code compiles Prints “m1 method of class A2”.
Why?

# What will be the result, when you try to compile and execute : (Contd.).

```
class A1 {
    protected void m1() {
        System.out.println("m1 method of class A1");
    }
}
class A2 extends A1 {
    void m1(int i) {
        System.out.println("m1 method of class A2");
    }
    public static void main(String[] args) {
       A2 x = new A2();
       x.m1();
    }
}
```

Code compiles and Prints "m1 method of class A1".
Why?

# Summary

In this session, you were able to learn about:

- Import
- Static Import
- Creating Our own packages

# Thank You

Sensitivity: Internal & Restricted