# Lecture5 - OOP concepts in Java

- Packages
- Exceptions

# Packages

# Packages

- A way of grouping different (related) classes in Java.
- Java itself provides many packages
  - E.g. Math, I/O, Exception, etc.
- Packages are used to provide
  - Access restrictions
  - Namespace management

# How to create packages

- Simply put "package" in the beginning of a class (should be the first line).

```
package example_package
 class myClass1
 {
            // Code
 }

package example_package
 class myClass2
 {
            // Code
 }
```

- myClass1 and myClass2 are now part of example_package
- A package typically has many classes.

# Creating packages example

```
package graphics;
public interface Draggable { . . . }

package graphics;
public abstract class Graphic { . . . }

package graphics;
public class Circle extends Graphic implements Draggable { . . . }

package graphics;
public class Rectangle extends Graphic implements Draggable { . . . }

package graphics;
public class Point extends Graphic implements Draggable { . . . }
```

Source: oracle.com

# Using classes from external packages

- Use **import** keyword.
  - Can import the entire package. E.g.,
    - import java.lang.*;
    - import mypackage.*;
  - Or, can import specific classes in a package
    - import mypackage.myclass;

- E.g. Use math functions.

```
import java.lang.math;
public class myClass
{
   public double computeArea (int r)
   {
            return ( math.PI * r * r);
   }
}
```

# Packages … contd.

- Packages can be created, included in a hierarchical way
  - E.g., com. mycompany.mypackage
    - Package from mycompany
  - com.anothercompany.package
    - Package from anothercompany.
  - They can be included as
    - import com.mycompany.mypackage
    - import com.anothercompany.mypackage

# Exceptions

# Exceptions

- A way to handle error or unexpected conditions.
- Used to ensure that error conditions are handled gracefully.
- In Java, there is a class called Exception that is used to handle any generic exceptional condition.
  - Exception is derived from Throwable
- Many kinds of specific exceptions are also available
  - I/O exceptions
  - Array out of bound exceptions
  - Class not found exception
  - No such method exception
- Users can define their own exceptions
  - Derive their class from Exception

# How to catch exceptions

- Use try catch statements

```
try
  {
    // some code
   }
catch (AnyException e)
  {
    // Error handling code
  }
```

- AnyException is any Exception
  - Can be Java defined exception, or user defined

# Throwing Exceptions

- Throw an exception using

  throw (e);

- Any exception – user defined or Java defined exception (e) can be thrown.

# Throwing exceptions … contd.

- The function that throws any exception should define it.

```
void function1( ) throws AnyException
    {
        // code
                // in case of error conditions
        throw (new AnyException( ) );
    }
```

- Any function calling myFunction should catch the exception

```
void function2 ( )
    {
        try
            {
                function1( )
            }
        catch (AnyException e) // Or parent of AnyException
            {
                // Error handling code
            }
    }
```

# Java – finally

- finally – a way to handle any left over (cleanup) issues.
- Should be present in the end, after try and catch are done.
- Typically used to clean up resources, open files, file descriptors, sockets, etc.