Fundamentos de Java

Módulo 1 - Clase #05
Java Programmer SE 8

Paquetes

- Organización de clases.
- Evitan conflictos entre nombres de clases.
- Limitan acceso a sus clases.
- Utiliza la palabra reservada “package” seguido del nombre del paquete.
  - Usar minúsculas.
  - Usar nombres cortos para uno mejor identificación.
- Estructura:
  - paquete.clase / paquete.subpaquete.clase
  - paquete.clase.metodo / paquete.subpaquete.clase.metodo

Import

- Se utiliza la palabra reservada “import” para utilizar la funcionalidad de un paquete externo.
- “import” debe ser utilizada sobre la declaración de la clase.
- No utiliza espacio en memoria.
- Usos:
  - Específico
  - Generico (*)
- Import estático:
  - import static paquete.clase.metodo

Import Estático

```java
package com.gm; //Definición del paquete

public class Utileria {
    public static void imprimir(String s){
        System.out.println("Imprimiento mensaje: " + s);
    }
}
```

```java
//Importamos el método estático a utilizar
import static com.gm.Utileria.imprimir;

public class EjemploPaquetes {

    public static void main(String[] args) {
        imprimir("Hola");
    }
}
```

Paquetes más importantes en Java

Paquete	Contenido
java.lang	Contiene clases esenciales, se importa implícitamente sin necesidad de un sentencia import
java.util	Contiene las clases de utilería más comunes (Scanner).
java.io	Clases que definen distintos flujos de datos (input/output).
java.net	Se usa en combinación con las clases del paquete java.io para leer y/o escribir datos en la red.
java.applet	Contiene las clases necesarias para crear applets y se ejecutan en la ventana del navegador.
java.awt	Contiene clases para crear una aplicación GUI (Graphic User Interface) independiente de la plataforma.

Ejemplo

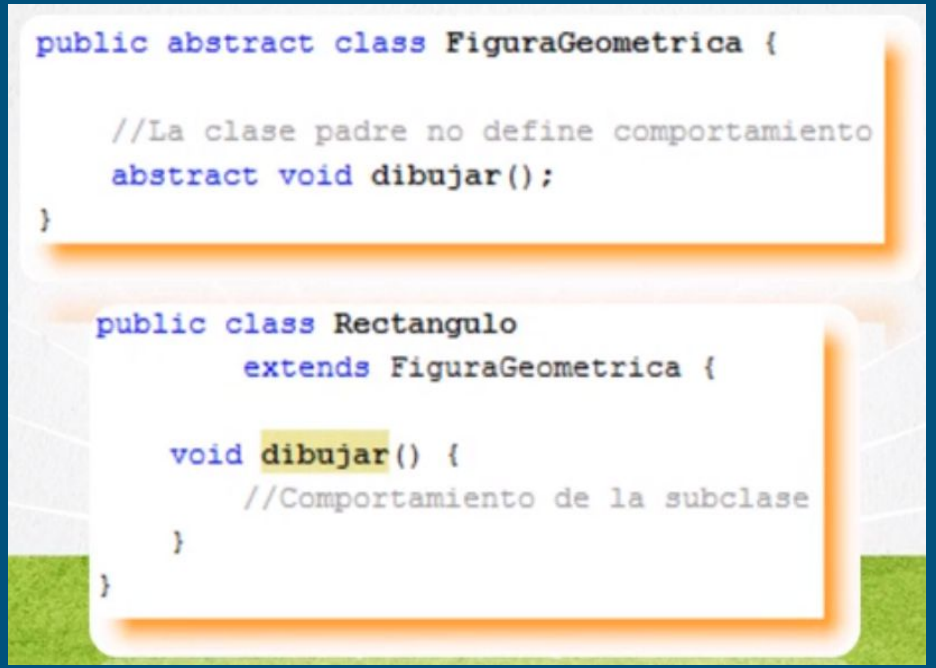
Clases Abstractas

Características:

- Definir una estructura.
- Aplicar el polimorfismo.
- Uso de la palabra reservada “extends”.

Restricciones:

- No se pueden instanciar.
- No definen comportamiento.
- La funcionalidad la establece las clases hijas.



```
public abstract class FiguraGeometrica {

    //La clase padre no define comportamiento
    abstract void dibujar();
}
```

```
public class Rectangulo
        extends FiguraGeometrica {

    void dibujar() {
        //Comportamiento de la subclase
    }
}
```

Ejemplo

Interfaces

- Declaración formal de un contrato.
- Son abstractos, es decir, no poseen implementación.
- Uso de la palabra reservada “implements”.
- Podemos implementar múltiples interfaces a una clase.

Estructura de una Interface

Definición de una interface en Java:

```
<modificadores>  interface <nombre_interface> [extends <interface padre>]
{
        <atributos>
        <métodos>
}
```

Uso de una interface en Java:

```
<modificadores> class <nombre_clase> [extends <superclase>] [implements
<interface1,interface2,etc>]
{
        <implementar_métodos_interface>
}
```

Ejemplo

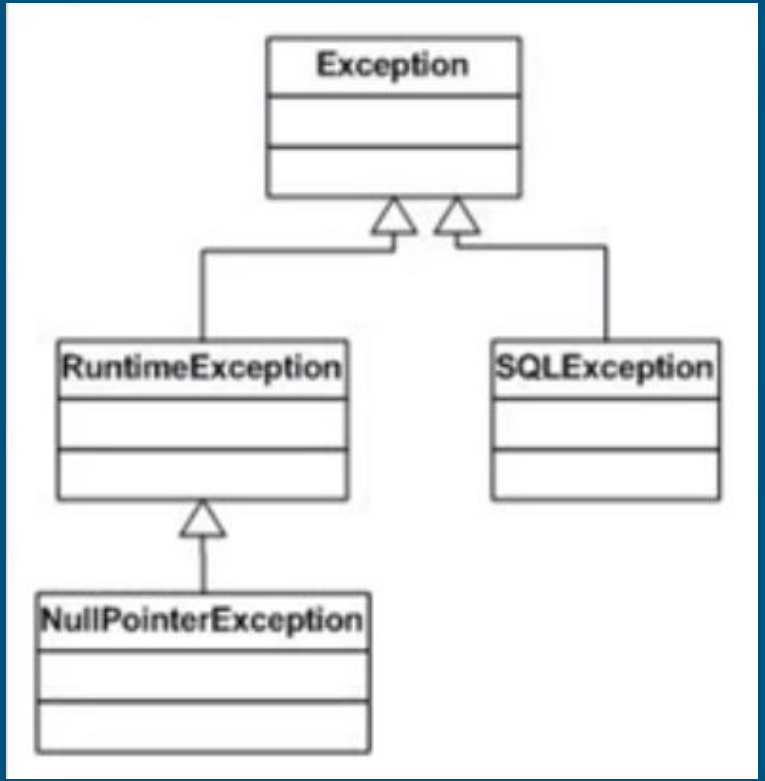
Excepciones

Check Exception

- Hereden de la clase Exception.
- Es obligatorio tratarlas.
- SQLException.

Unchecked Exception

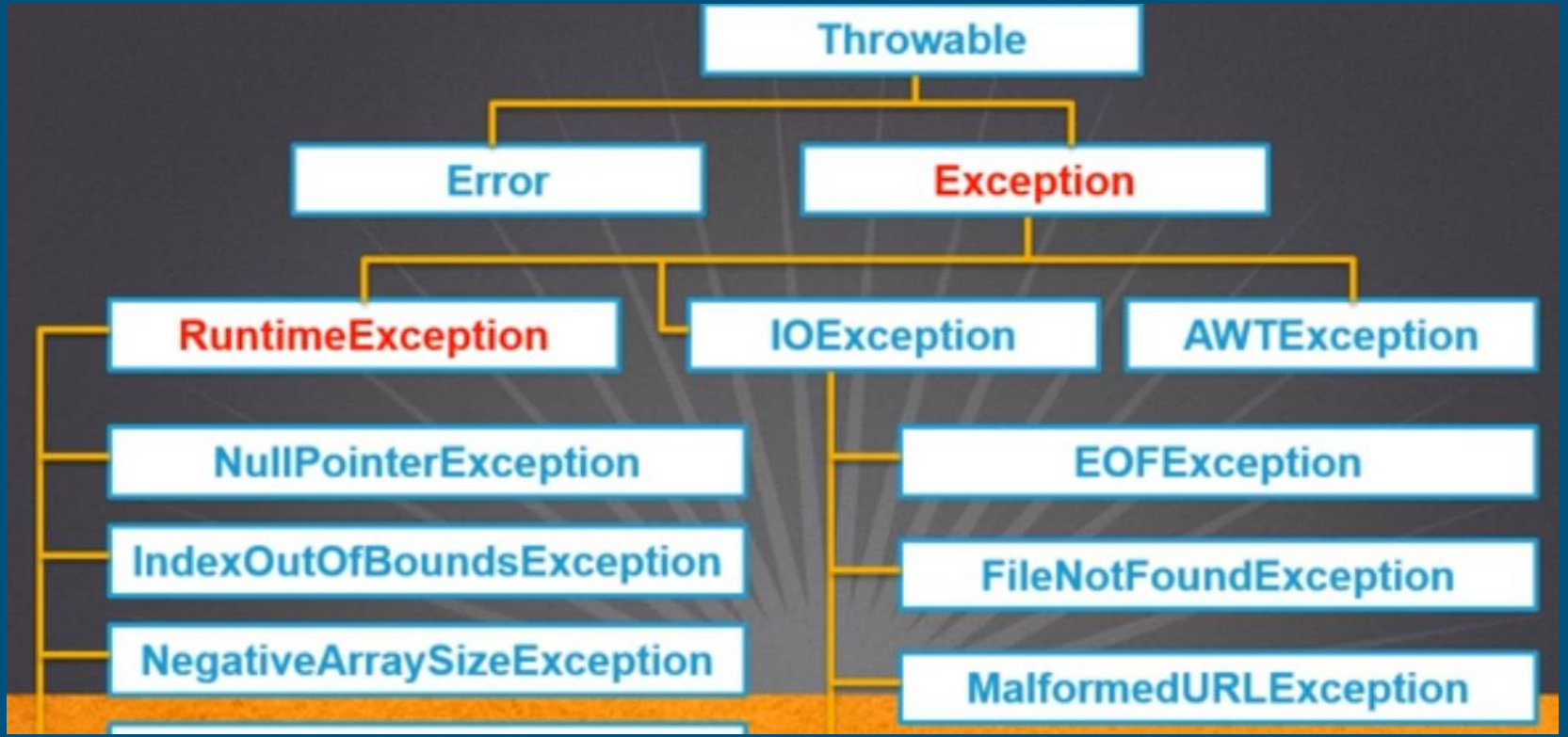
- Hereden de la clase Exception.
- Es opcional tratarla.
- RuntimeException.
    - NullPointerException.



Manejo de Excepciones

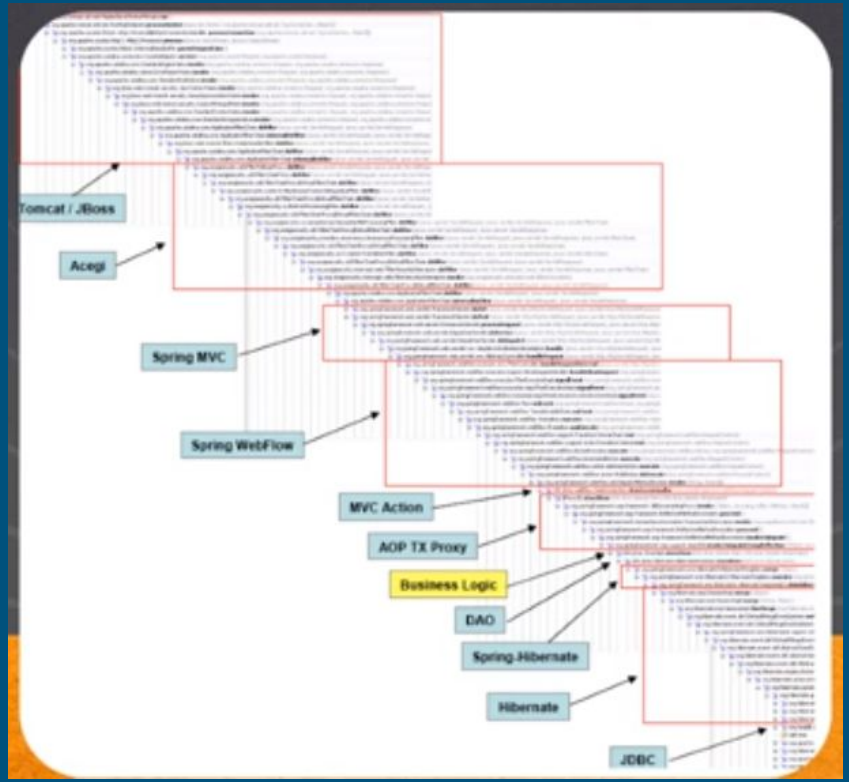
```
public void verificaExcepciones() {
   try {
      // código que lanza excepciones
   } catch (Exception ex) {
      //Bloque de código que maneja la excepción
      ex.printStackTrace();
   }
   finally{
      //Bloque de código opcional, pero
      //que se ejecuta siempre
   }
}
```

Excepciones más comunes



Stacktrace en Java

- Pilas de errores.
- Traza del error del inicio al fin.
- Flujo del error:
  - Método arroja un error.
  - Si no atrapa la excepción, la propaga hasta que alguna clase la maneja.
  - En caso de no manejarla el método main se satura
  - El programa se finaliza de manera anormal.



Cláusula Throws

```java
public class ArrojarExcepcion {

  public void metodoX() throws Exception {
    throw new Exception("Mensaje de error");
  }
}
```

```java
public class TestArrojarExcepcion {

  public static void main(String args[]) throws Exception {
    ArrojarExcepcion ae = new ArrojarExcepcion();
    ae.metodoX();
  }
}
```

Creación de Nuestras Excepciones

```
public class MiExcepcion extends Exception{

  public MiExcepcion(String mensaje){
    super(mensaje);
  }
}
```

```
public class ArrojarExcepcion2 {

  public void metodoX() throws MiExcepcion {
    throw new MiExcepcion("Mi mensaje de error");
  }
}
```

Ejemplo