Excepciones

Excepciones

```
class Ejercicio1{
    public static void main(String args[]){
        int[] arreglo=new int[2];
        arreglo[2]=0;
    }
}
```

Compila sin errores.

Al ejecutar da **ArrayIndexOutOfBoundsException**

```
class Ejercicio2{
    public static void main(String args[]){
        String a = null;
        a.trim();
    }
}
```

Compila sin errores.

Al ejecutar da **NullPointerException**

Excepciones

```
class Ejercicio3{
    public static void main(String args[]){
        File f=new File("archivo1.txt");
        f.createNewFile();
    }
}
```

ERROR DE COMPILACIÓN!!

Produce un error de **COMPILACIÓN, porque no se está tratando la Excepción del tipo IOException**

Excepciones

Porqué en un caso da ERROR DE COMPILACIÓN y en el otro no?

```
class Ejercicio3{
    public static void main(String args[]){
        File f=new File("archivo1.txt");
        f.createNewFile();//IOException
    }
}
```

```
class Ejercicio1{
    public static void main(String args[]){
        int arreglo=new int[2];
        arreglo[2]=0;  //ArrayIndexOutOfBoundsException
    }
}
```

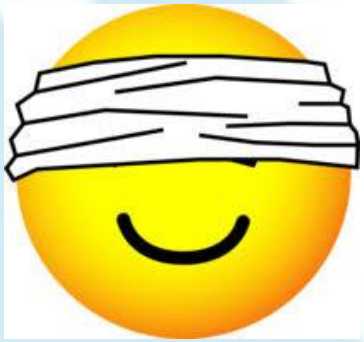
Por el tipo de Excepción

Excepciones

CHECKED

```
class Ejercicio3{
    public static void main(String args[]){
        File f=new File("archivo1.txt");
        f.createNewFile();//IOException
    }
}
```

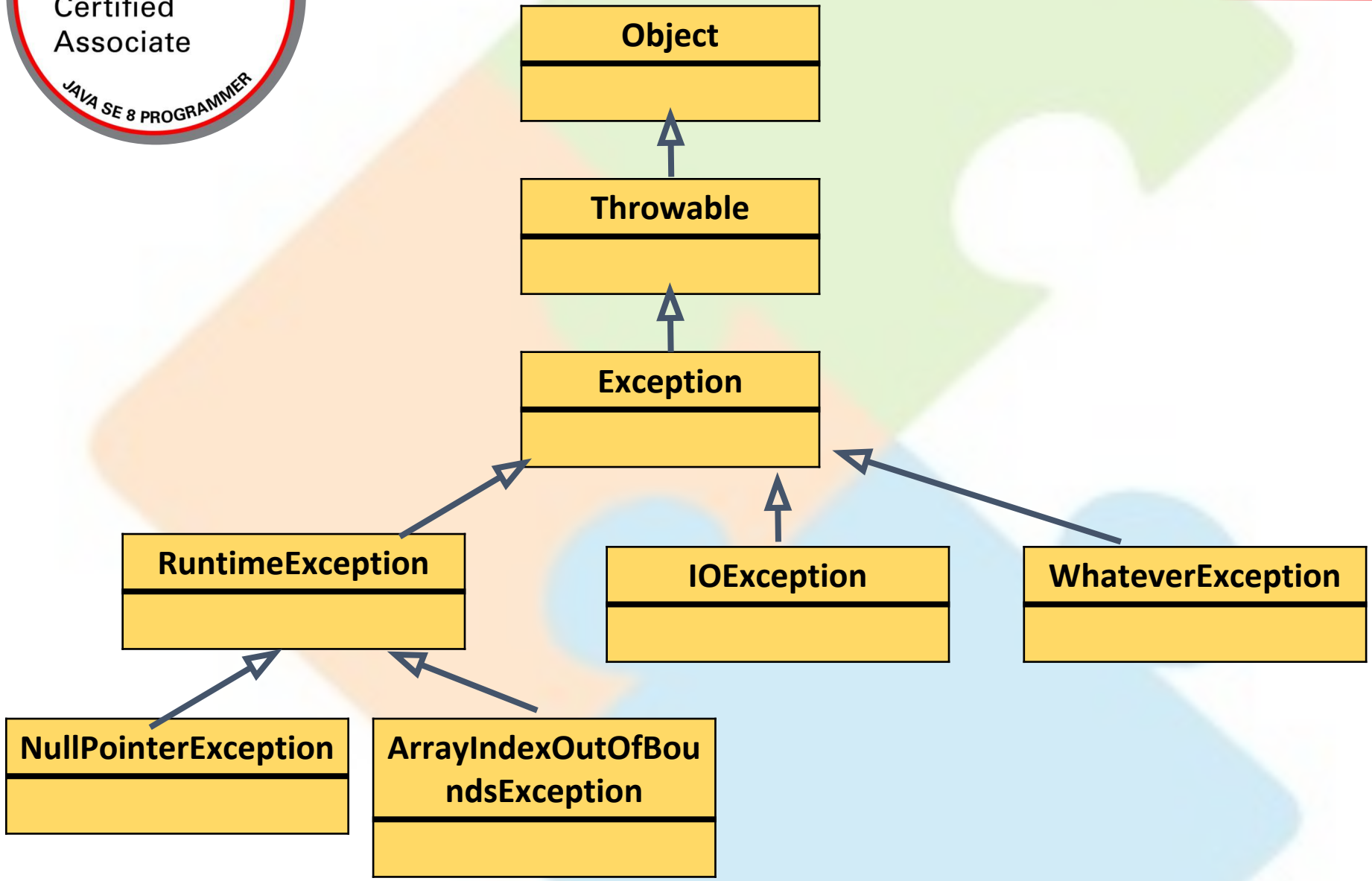
UNCHECKED



```
class Ejercicio1{
    public static void main(String args[]){
        int arreglo=new int[2];
        arreglo[2]=0;
//ArrayIndexOutOfBoundsException
    }
}
```

Las CHECKED son las que molestan en compilación en cada punto.

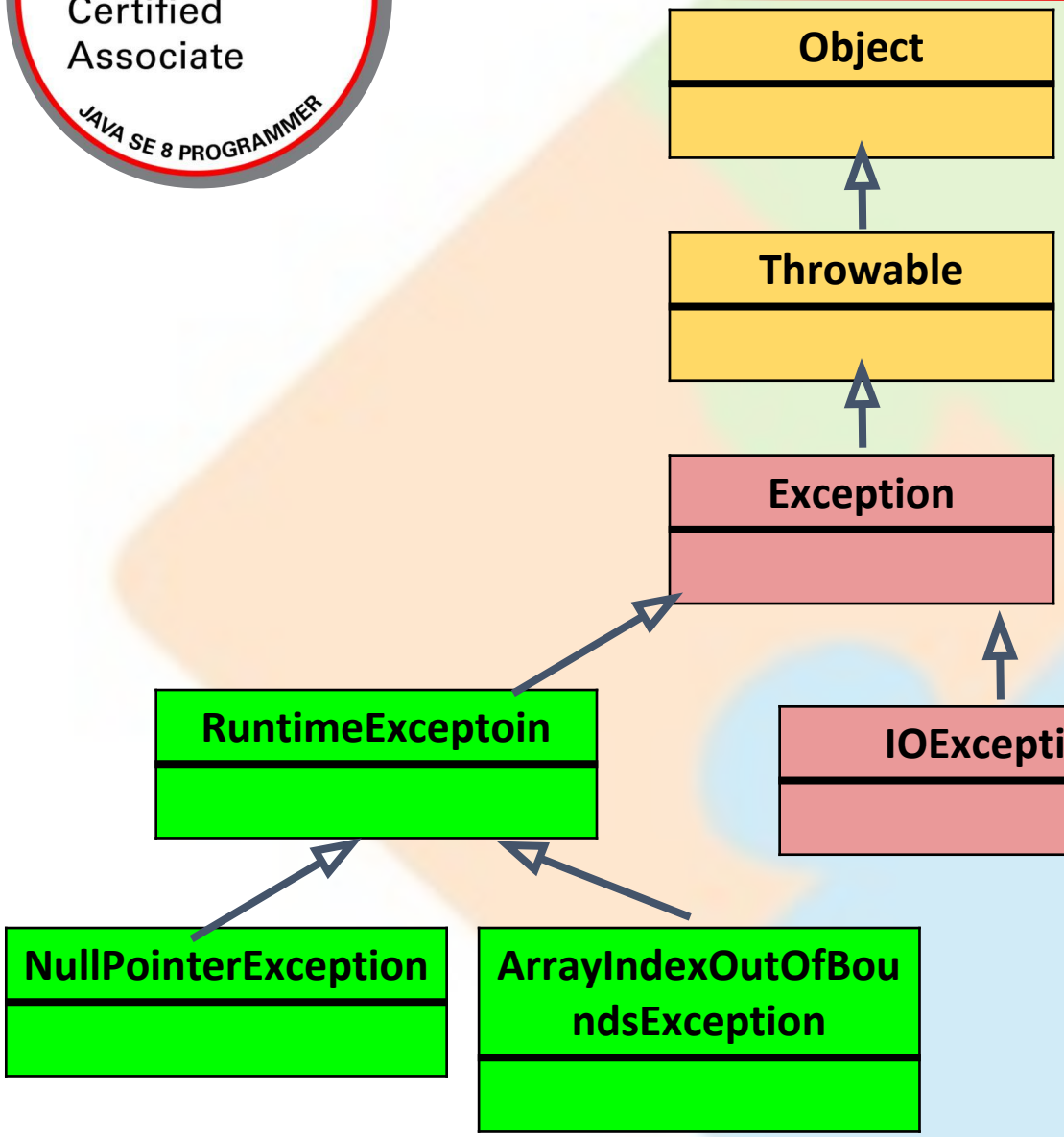
Excepciones



Excepciones

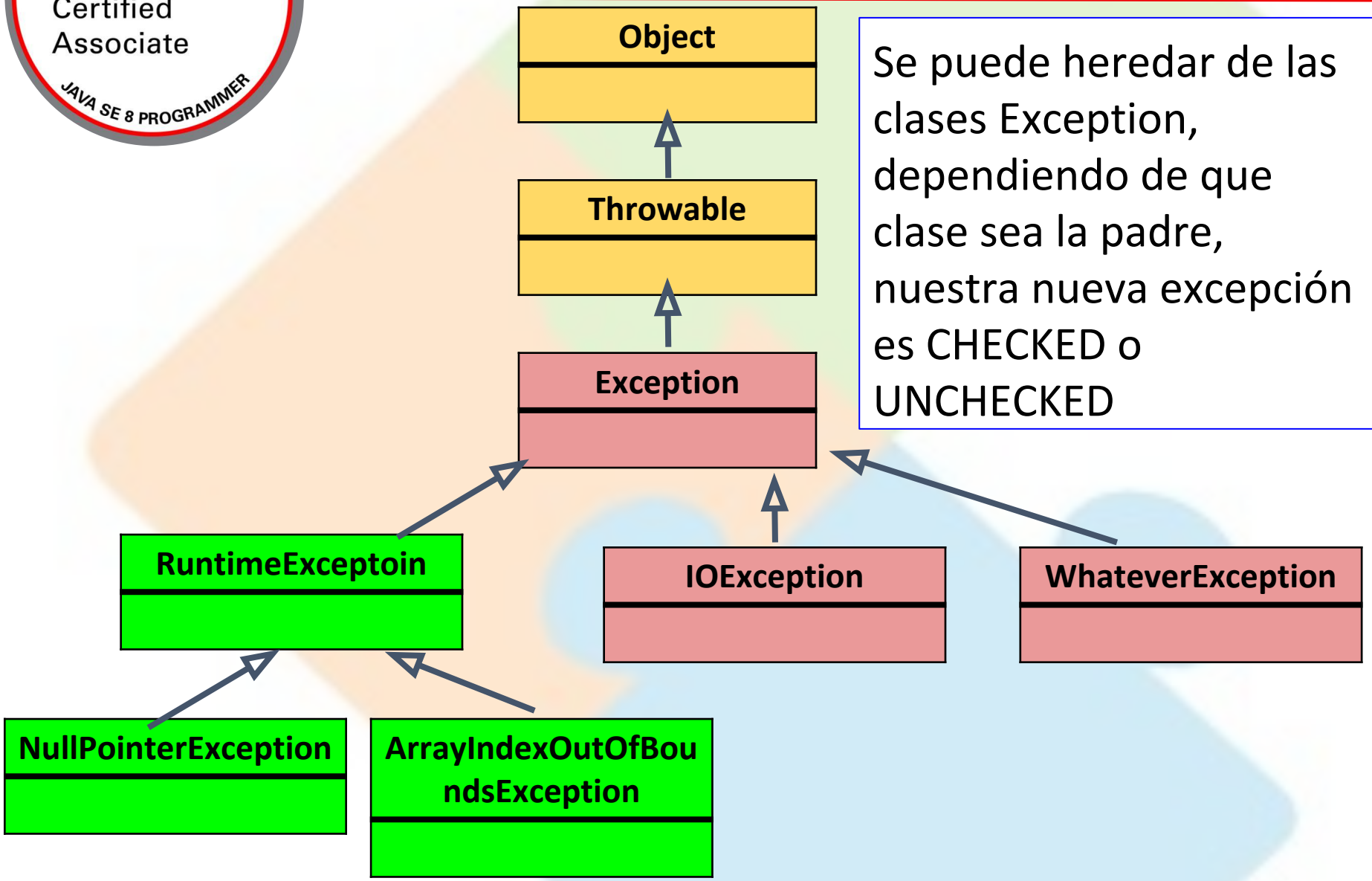
RuntimeExcep
todas sus hijas son
UNCHECKED

Excepciones



Todas las que NO SON RuntimeException son CHECKED

Excepciones

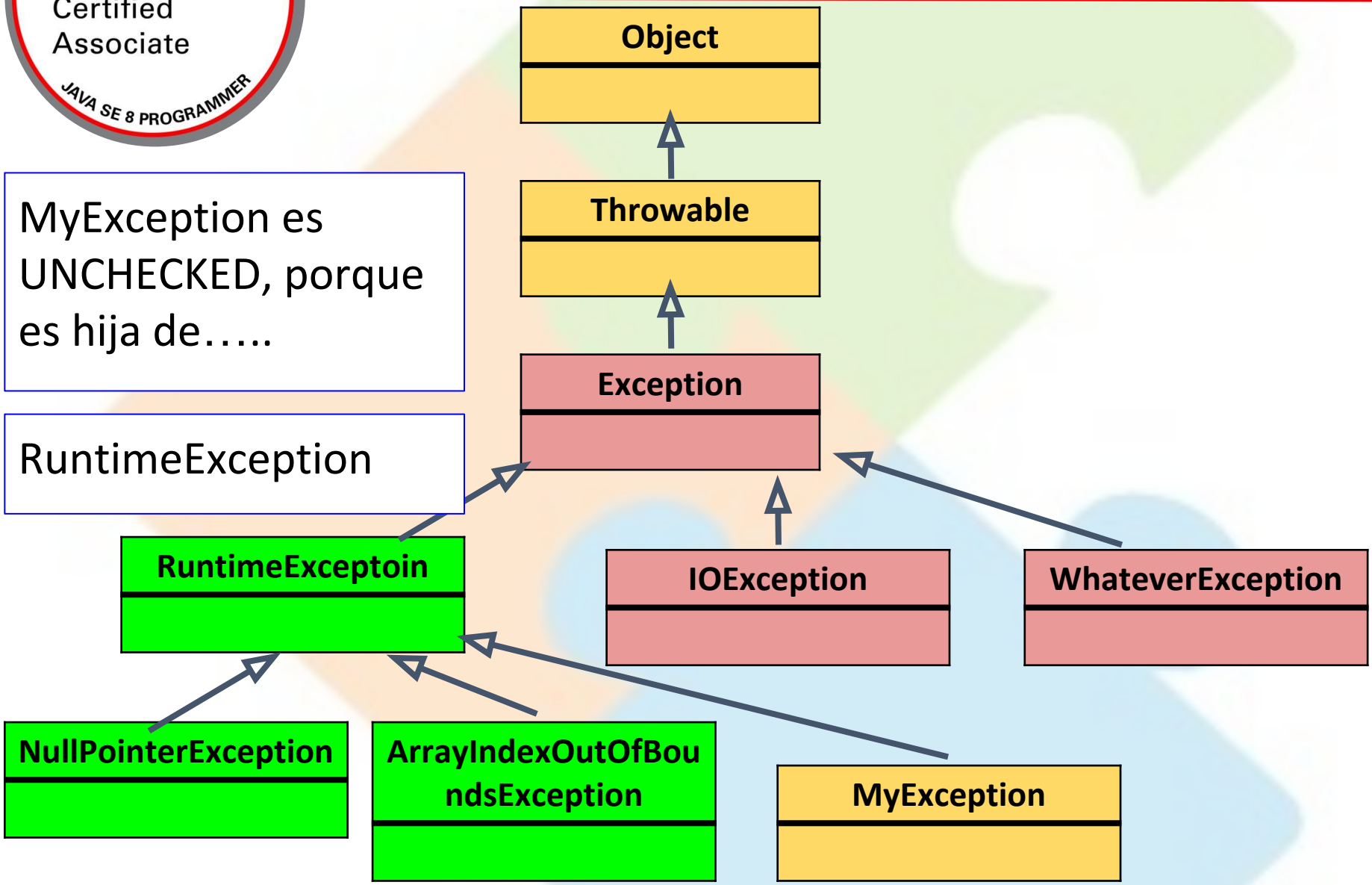


Se puede heredar de las clases Exception, dependiendo de que clase sea la padre, nuestra nueva excepción es CHECKED o UNCHECKED

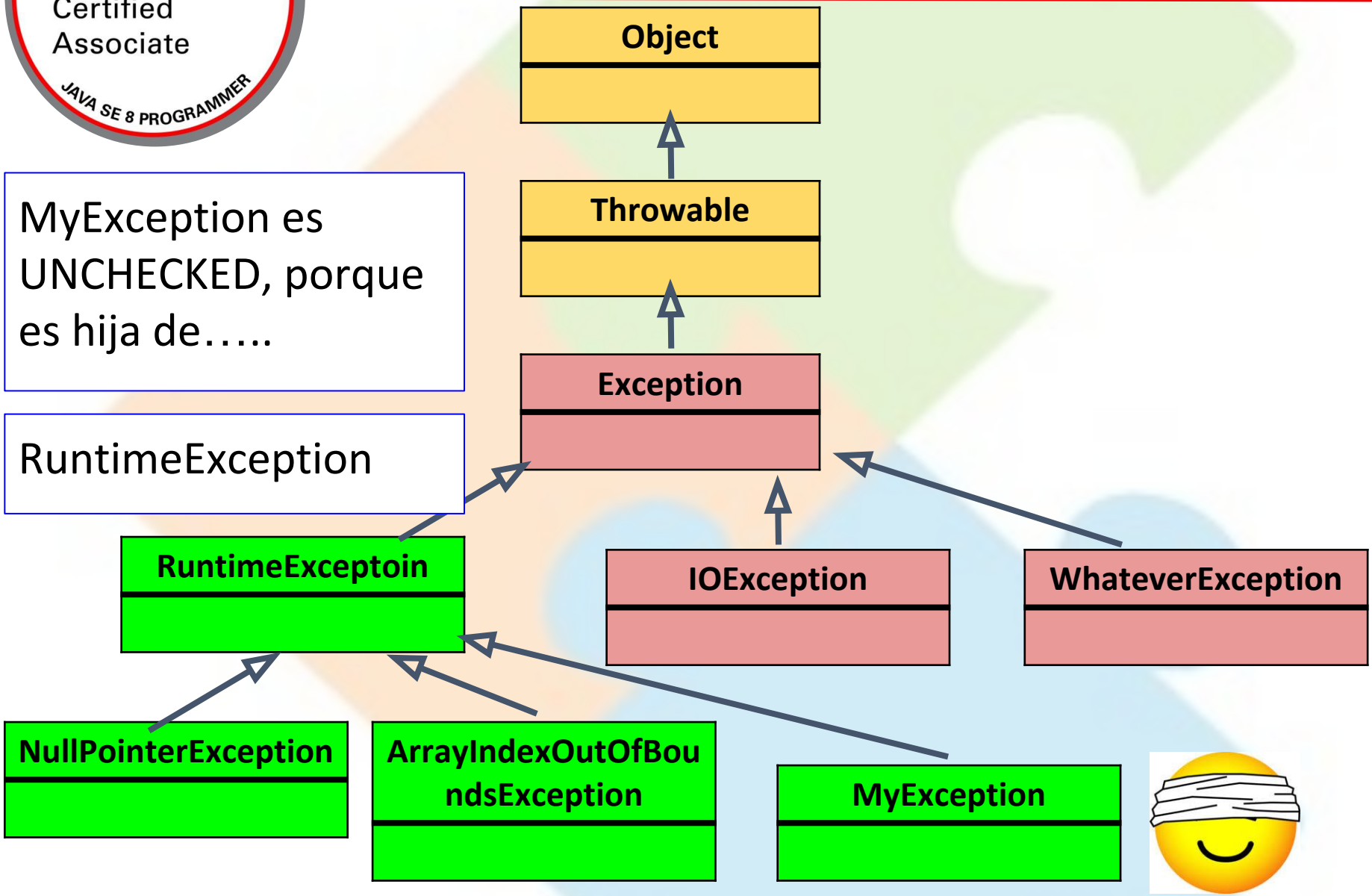
Excepciones

MyException es UNCHECKED, porque es hija de…..

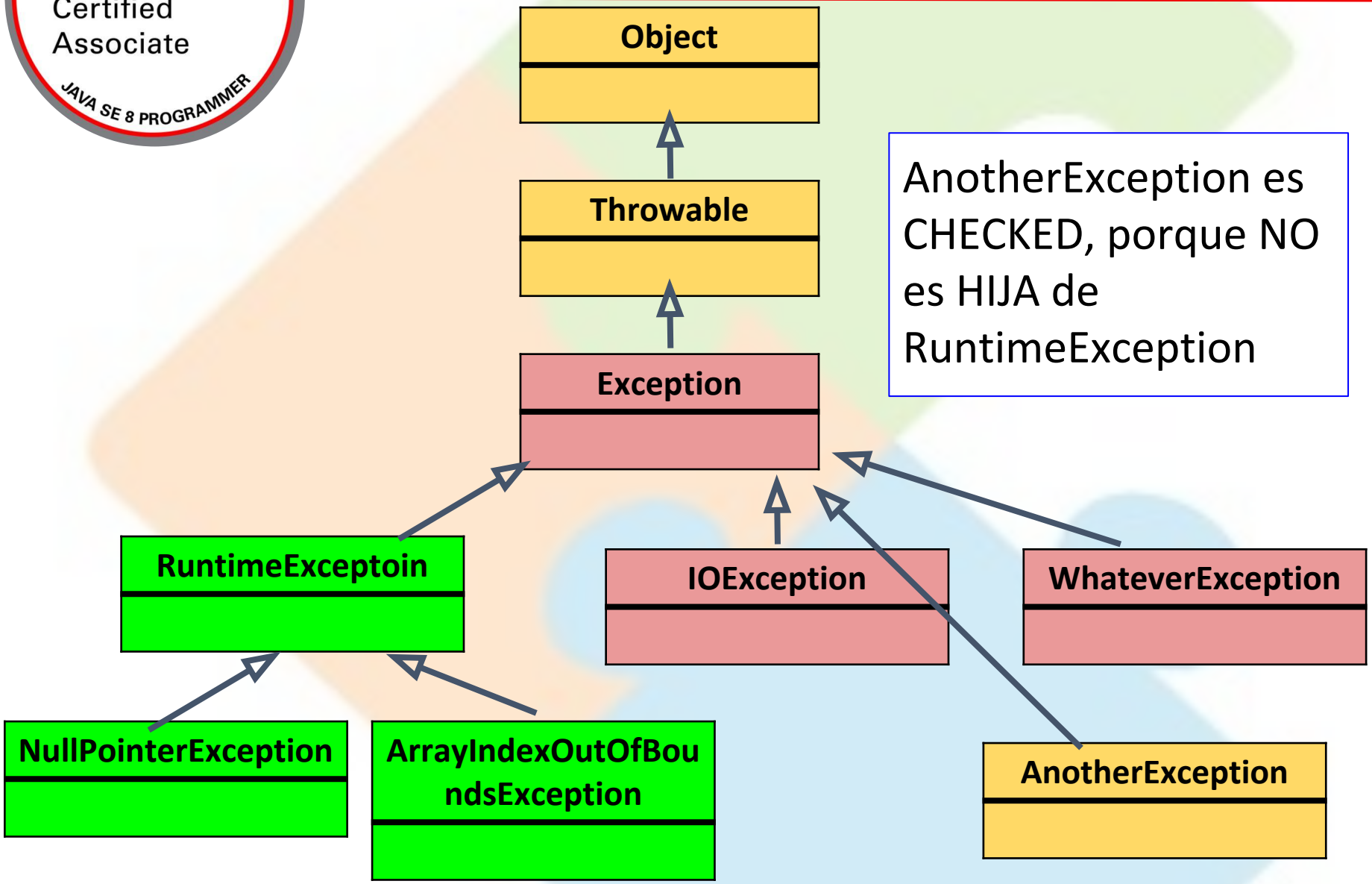
RuntimeException



Excepciones

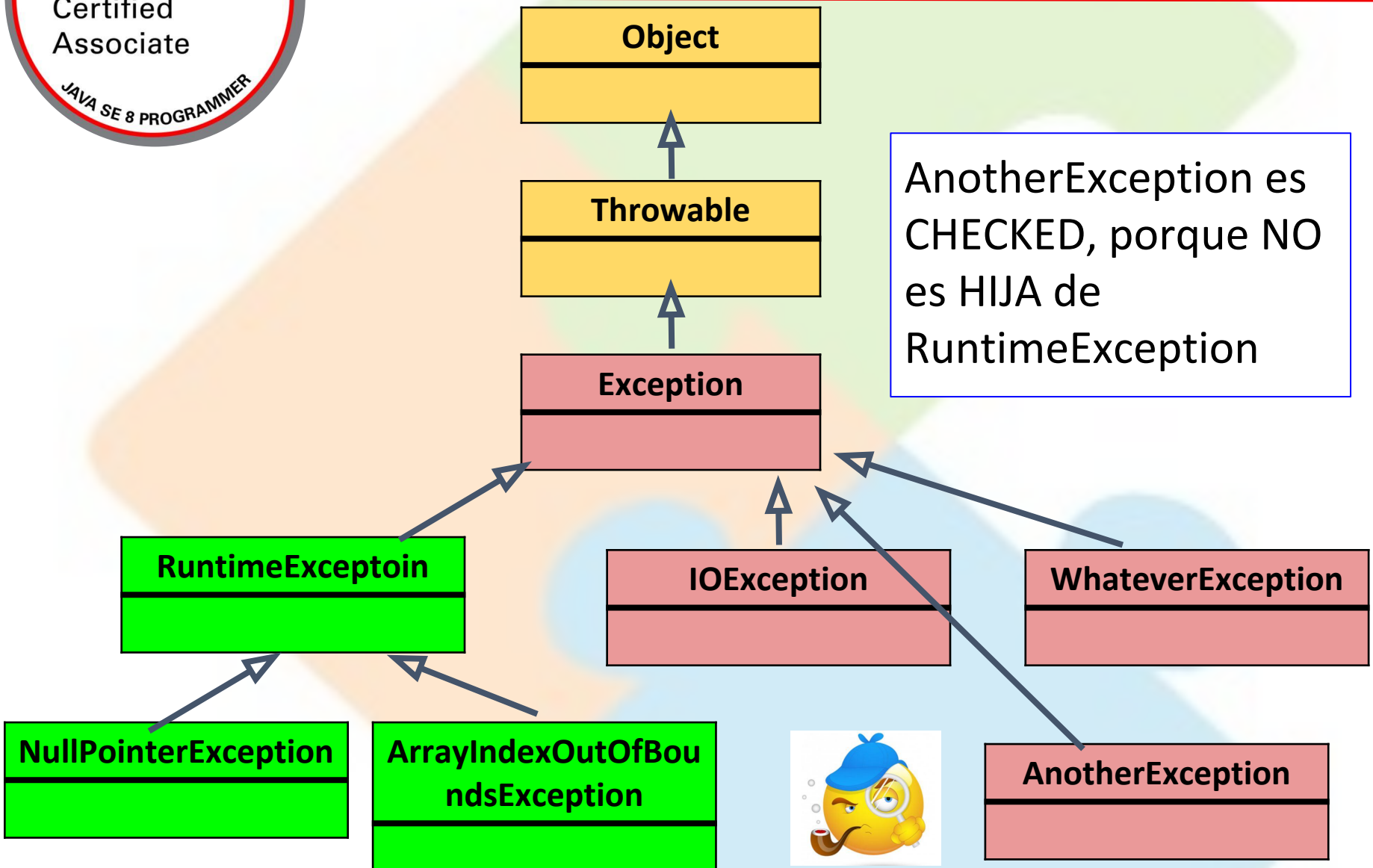


Excepciones



AnotherException es CHECKED, porque NO es HIJA de RuntimeException

Excepciones



Excepciones

Podemos crear nuestras propias Excepciones heredando de Exception o sus hijas

```
public class MyException extends Exception{ }
```

```
public class AnotherException extends
RuntimeException{ }
```

Excepciones

Desde un método se pueden arrojar excepciones usando **throw** y el objeto Exception que se quiere lanzar

```
public void metodo1(){
    throw new Exception();
}
```

```
public void metodo2(){
    throw new RuntimeException();
}
```

Desde un método se pueden arrojar excepciones usando **throw** y el objeto Exception que se quiere lanzar

```
public void metodo1(){
    throw new Exception();
}
```

ERROR DE COMPILACIÓN!!

```
public void metodo2(){
    throw new RuntimeException();
}
```

Desde un método se pueden arrojar excepciones usando **throw** y el objeto Exception que se quiere lanzar

```
public void metodo1(){
    throw new Exception();
}
```

ERROR DE COMPILACIÓN!!

Si la Excepción es CHECKED se debe colocar la Excepción en la firma del método

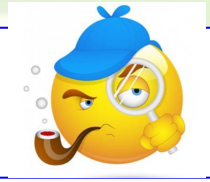
Desde un método se pueden arrojar excepciones usando **throw** y el objeto Exception que se quiere lanzar

```
public void metodo1() throws Exception{
    throw new Exception();
}
```

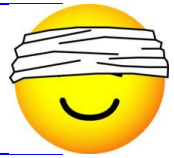
Si la Excepción es **CHECKED** se debe colocar la Excepción en la firma del método con la palabra **throws**

Excepciones

```
public void metodo1() throws Exception{  }
```



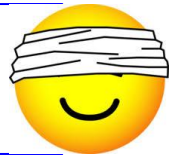
```
public void metodo2() throws RuntimeException{  }
```



```
public void metodo3(){
    metodo1();
    metodo2();
}
```

```
public void metodo1() throws Exception{ }
```

```
public void metodo2() throws RuntimeException{ }
```



Cuando se invoca a un método que tiene en su firma una excepción CHECKED, da error de compilación, a menos que se TRATE la Excepción

```
public void metodo3(){
    metodo1();
    metodo2();
}
```

ERROR DE COMPILACIÓN!!

Excepciones

```
public void metodo1() throws Exception{  }
```

```
public void metodo3(){
    metodo1();
}
```

ERROR DE COMPILACIÓN!!

Para tratar la Excepción existen 2 formas:

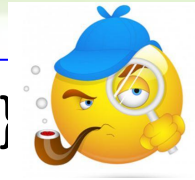
Envolverle en un try/catch

ó

Propagar la Excepción (lavarse las manos, pasar la pelotita

Excepciones

```
public void metodo1() throws Exception{  }
```

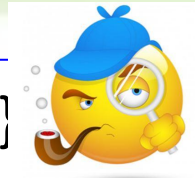


Envolverle en try/catch

```
public void metodo3(){
  try{
       metodo1();
   }catch(Exception ex){}
}
```

Excepciones

```
public void metodo1() throws Exception{  }
```

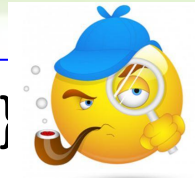


Propagar la Excepción

```
public void metodo3() throws Exception{
    metodo1();
}
```

Excepciones

```
public void metodo1() throws Exception{  }
```



Propagar la Excepción

```
public void metodo3() throws Exception{
    metodo1();
}
```

Excepciones

En las excepciones CHECKED el compilador nos exige colocarlas en la firma del método, también tratarlas.

En las UNCHECKED, no nos exige nada pero si queremos lo podemos hacer

Excepciones

```
public class MyException extends Exception{ }
```

Por Herencia:
MyException **ES UNA** Exception

Excepciones

```
public class MyException extends Exception{ }
```

```
public void metodo1() throws MyException{
    throw new MyException();
}
```

Compila porque MyException **ES UNA** Exception, entonces throws Exception incluye al tipo MyException

```
public void metodo2() throws Exception{
    throw new MyException();
}
```

Excepciones

```
public void metodo1() throws Exception{
   double x=Math.random()*10;
   Exception ex=(x<5)?new Exception():new RuntimeException();
   throw ex;
}
```

Excepciones

```
public void metodo1() throws Exception{
   double x=Math.random()*10;
   Exception ex=(x<5)?new Exception():new RuntimeException();
   throw ex;
}
```

```
public void metodo1() throws Exception{
   double x=Math.random()*10;
   throw (x<5)?new Exception():new RuntimeException();

}
```