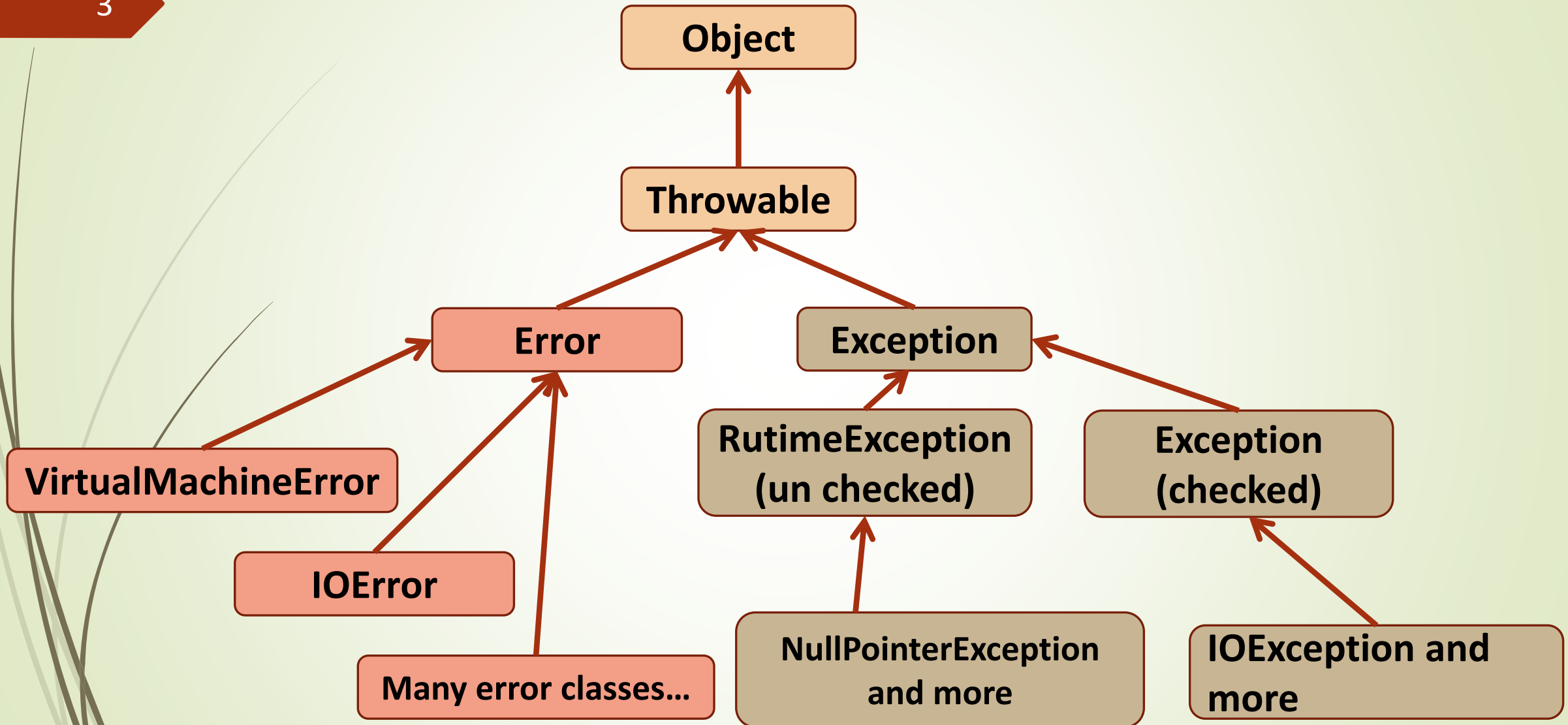
Exception Handling

Exceptions

- An **exception** is a abnormal condition that occurs at run time and disturbs normal continuation of the program
- When an exception occurs, the program must either terminate or jump to exception handling code
- The special code for exception handling the is called an **Exception Handler**
- **Exception Conditions: Divide by Zero, Array Out Of Bound etc.**

Exception Hierarchy in Java



Exceptions Handling - keywords

- **throw** – throw is used to throw the exception instead of handling it using **try** and **catch**
- **try** – try block is used to invoke code that may throw an exception
- **catch** – catch block is used to handle exceptions thrown in preceding try block.
  - **catch** block can not be written without **try block**

Exceptions Handling - keywords

- **throws** – throws is used to declare which exceptions can be thrown by methods

- **finally** – finally block is used to execute code for releasing or freeing resources.
  - finally block always survives(except System.exit(0) or JVM termination)

Unhandled Exceptions

- An unhandled exception propagates backwards into the calling method and appears to be thrown at the point of the call
- The JVM will keep terminating method calls and tracing backwards along the call chain until it finds an enclosing **try** block with a matching handler or catch, or until the exception propagates out of **main to JVM** (terminating the program).

Handling Multiple Exceptions

- Multiple catch blocks can be attached to the same block of code. The catch blocks should handle exceptions of different types

```
try{...}
 catch(...){ }
 catch(...){ }
 catch(...){ }
```

Generic catch block

- Java allow user to write generic catch block to handle all types of Exception/Error but its not good practice to catch all exceptions in one catch block

Example:

```
catch( Throwable t) {
//This catch block can handle any exception/error thrown
}
```

Custom Exception

- Programmer can create custom exception by inheriting any built-in Exception classes
- Ex. *CustomException*

```
public class CustomException extends Exception {
   public CustomException(String message)  {
    super(message);
   }
}
```

try-with-resources

- Java 7 feature - Java has introduced java.lang.AutoCloseable – Interface
- Resources that must be closed need be declared in try
- Autocloesable interface has close method (public void close() throws Exception)
- syntax of try-with-resources

```
try (//1 or more AutoCloseable resources){
//Code
} catch(Exception e) {
//Handling code
}
```

- eg :

```
try(Scanner sc=new Scanner(System.in);){
 //Code
} catch(Exception e) {
//handling code}
```