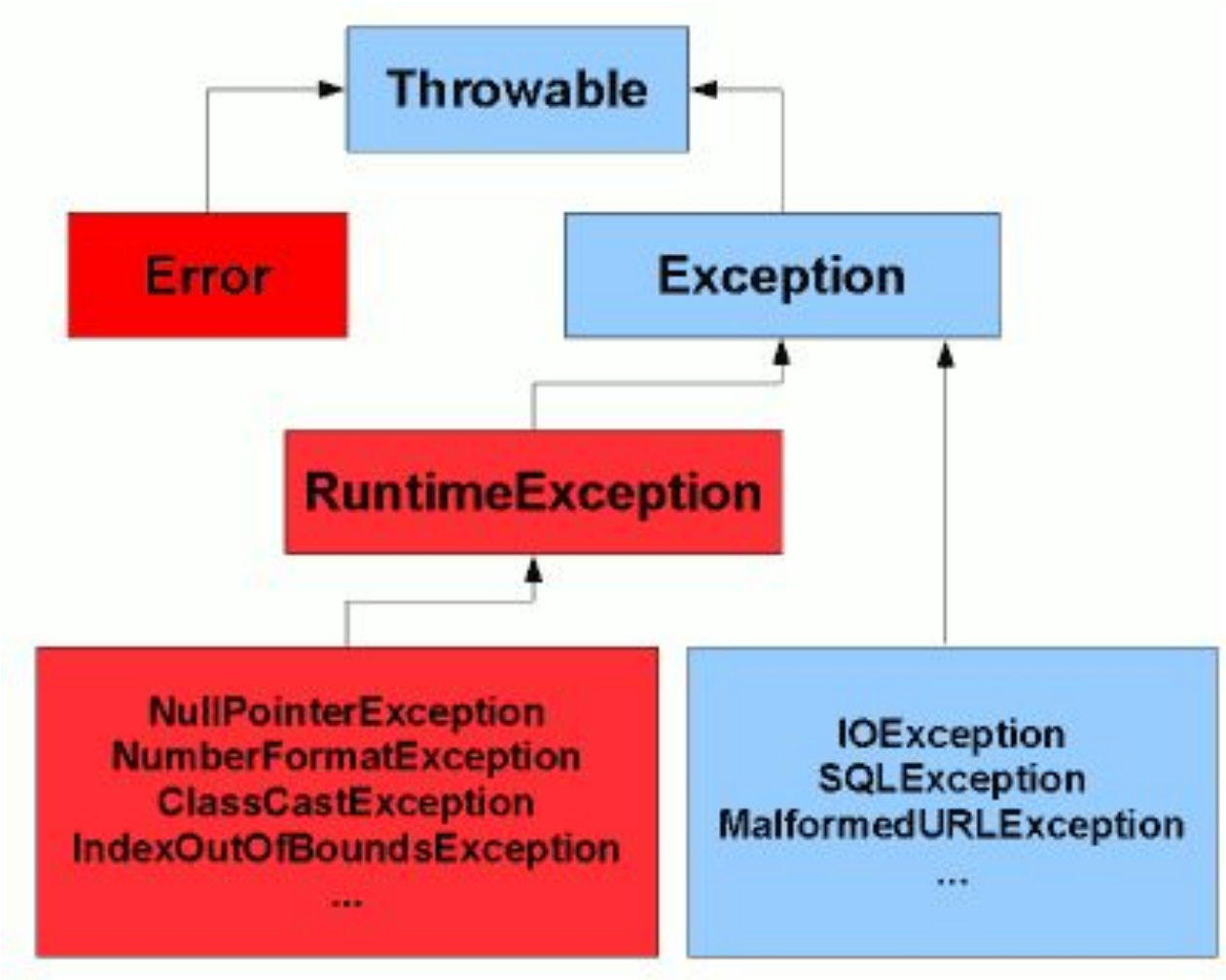
Exceptions

Исключение в Java - это объект, который описывает исключительное состояние, возникшее в каком-либо участке программного кода.

Summary

1. **Классификация исключений**
2. Инициация и обработка исключений
3. Блок finally
4. Подводные камни
5. Java SE 7 features
   - 5.1 Multi-catch exceptions
   - 5.2 Rethrowing exceptions
   - 5.3 Try-with-Resources

Классификация исключений



Class Exception

- java.lang.Object
  - java.lang.Throwable
    - java.lang.Exception
- **All Implemented Interfaces:**
  - Serializable
- **Direct Known Subclasses:**
  - AclNotFoundException, ActivationException, AlreadyBoundException, ApplicationException, AWTException, BackingStoreException, BadAttributeValueExpException, BadBinaryOpValueExpException, BadLocationException, BadStringOperationException,BrokenBarrierException, CertificateException, CloneNotSupportedException, DataFormatException, DatatypeConfigurationException, DestroyFailedException, ExecutionException, ExpandVetoException, FontFormatException, GeneralSecurityException, GSSException,IllegalClassFormatException, InterruptedException, IntrospectionException, InvalidApplicationException, InvalidMidiDataException, InvalidPreferencesFormatException, InvalidTargetObjectTypeException, IOException, JAXBException, JMException,KeySelectorException, LastOwnerException, LineUnavailableException, MarshalException, MidiUnavailableException, MimeTypeParseException, MimeTypeParseException, NamingException, NoninvertibleTransformException, NotBoundException...

Class RuntimeException

- java.lang.Object
  - java.lang.Throwable
    - java.lang.Exception
      - java.lang.RuntimeException
- **All Implemented Interfaces:**
  - Serializable
- **Direct Known Subclasses:**
  - AnnotationTypeMismatchException, ArithmeticException, ArrayStoreException, BufferOverflowException, BufferUnderflowException, CannotRedoException, CannotUndoException, ClassCastException, CMMException, ConcurrentModificationException,DataBindingException, DOMException, EmptyStackException, EnumConstantNotPresentException, EventException, FileSystemAlreadyExistsException, FileSystemNotFoundException, IllegalArgumentException, IllegalMonitorStateException, IllegalPathStateException,IllegalStateException, IllformedLocaleException, ImagingOpException, IncompleteAnnotationException, IndexOutOfBoundsException, JMRuntimeException, LSException, MalformedParameterizedTypeException, MirroredTypesException, MissingResourceException,NegativeArraySizeException, NoSuchElementException, NoSuchMechanismException, NullPointerException, ProfileDataException, ProviderException...

Class Error

- java.lang.Object
  - java.lang.Throwable
    - java.lang.Error
- **All Implemented Interfaces:**
  - Serializable
- **Direct Known Subclasses:**
  - AnnotationFormatError, AssertionError, AWTError, CoderMalfunctionError, FactoryConfigurationError, FactoryConfigurationError, IOError, LinkageError, ServiceConfigurationError, ThreadDeath, TransformerFactoryConfigurationError, VirtualMachineError

1. Классификация исключений
2. **Инициация и обработка исключений**
3. Блок finally
4. Подводные камни
5. Java SE 7 features
   - 5.1 Multi-catch exceptions
   - 5.2 Rethrowing exceptions
   - 5.3 Try-with-Resources

throw

Оператор **throw** используется для выбрасывания исключения «вручную». Для того, чтобы сделать это, нужно иметь объект подкласса класса Throwable, который можно либо получить как параметр оператора catch, либо создать с помощью оператора new.

```
try {
    throw new NullPointerException("demo");
}
catch (NullPointerException e) {
    System.out.println("перехватили исключение: " + e);
    throw e;
}
```

Инициация исключений

1)

```
throw IllegalArgumentException(“Текстовое сообщение”);
```

2)

```
catch (NullPointerException e) {
    throw MyException(“Текстовое сообщение”, e);
}
```

Обработка исключений

```
catch (Exception e) {
    System.out.println(e);
}
```

```
catch (Exception ex) {
    //обработка исключения
}
catch (IOException ioex) {
    //обработка исключения
}
```

1. Классификация исключений
2. Инициация и обработка исключений
3. **Блок finally**
4. Подводные камни
5. Java SE 7 features
   - 5.1 Multi-catch exceptions
   - 5.2 Rethrowing exceptions
   - 5.3 Try-with-Resources

Блок finally из конструкции try/catch/finally

Bad code:

```
try {
   Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/db");
   Statement stmt = conn.createStatement();
   ResultSet rs = stmt.executeQuery("select * from my_table");
   // do something
   stmt.close();
   conn.close();
} catch (SQLException ex) {
   // handle exception
}
```

Блок finally из конструкции try/catch/finally

Good code:

```
try {
   // --//--
   ResultSet rs = stmt.executeQuery("select * from my_table");
   // do something
} catch (SQLException ex) {
   // handle exception
} finally {
    if (conn  != null) {
        try {
           conn.close();
        } catch (SQLException e1) {
           //do something
         }
    }
}
```

Пример

```
public class FinallyTest{
    public static int stringSize(Object s) {
        try {
            return s.toString().length();
        } catch (Exception ex) {
            return -1;
        } finally {
            return 0;
        }
    }
    public static void main(String[] args){
        System.out.println("stringSize(\"string\"):"
            +stringSize("string"));
        System.out.println("stringSize(null):"
            +stringSize(null));
    }}
```

stringSize("string"): 0
stringSize(null): 0

1. Классификация исключений
2. Инициация и обработка исключений
3. Блок finally
4. **Подводные камни**
5. Java SE 7 features
   - 5.1 Multi-catch exceptions
   - 5.2 Rethrowing exceptions
   - 5.3 Try-with-Resources

Блок finally может вызвать потерю исключений

```
public class ExceptionLossTest{
    public static void main(String[] args){
        try {
            try {
                throw new Exception("a");
            } finally {
                if (true) {
                    throw new IOException("b");
                } System.err.println("c");
            }
        } catch (IOException ex) {
            System.err.println(ex.getMessage());
        } catch (Exception ex) {
            System.err.println("d");
            System.err.println(ex.getMessage());
}   }   }
```

Отсутствие транзакционности

```
public class PartialInitTest{
    static PartialInitTest self;
    private int field1 = 0;
    private int field2 = 0;
    public PartialInitTest(boolean fail) throws Exception{
        self = this;
        field1 = 1;
        if (fail) {
            throw new Exception();
        }
        field2 = 1;
    }
    public boolean isConsistent(){
        return field1 == field2;
    }
```

Продолжение следует...

Отсутствие транзакционности (продолжение)

```
    public static void main(String[] args){
        PartialInitTest pit = null;
        try {
            pit = new PartialInitTest(true);
        } catch (Exception ex){
            // do nothing
        }
        System.out.println("pit: " + pit);
        System.out.println("PartialInitTest.self reference: " +
            PartialInitTest.self);
        System.out.println("PartialInitTest.self.isConsistent(): " +
        PartialInitTest.self.isConsistent());
    }
}
```

pit: null
PartialInitTest.self reference: test.PartialInitTest@1e0bc08
PartialInitTest.self.isConsistent(): false

Очень важная особенность исключений

Свойством транзакционности исключения не обладают - действия, произведенные в блоке try *до* возникновения исключения, не отменяются *после* его возникновения.

1. Классификация исключений
2. Инициация и обработка исключений
3. Блок finally
4. Подводные камни
5. **Java SE 7 features**
   - 5.1 **Multi-catch exceptions**
   - 5.2 Rethrowing exceptions
   - 5.3 Try-with-Resources

Multi-catch exceptions

Before Java SE 7:

```
catch (IOException ex) {
     logger.log(ex);
     throw ex;
catch (SQLException ex) {
     logger.log(ex);
     throw ex;
}
```

From Java SE 7:

```
catch (IOException|SQLException ex) {
    logger.log(ex);
    throw ex;
}
```

1. Классификация исключений
2. Инициация и обработка исключений
3. Блок finally
4. Подводные камни
5. **Java SE 7 features**
   - 5.1 Multi-catch exceptions
   - 5.2 **Rethrowing exceptions**
   - 5.3 Try-with-Resources

Rethrowing exceptions

```
static class FirstException extends Exception { }
static class SecondException extends Exception { }
```

Before Java SE 7:

```
  public void rethrowException(String exceptionName) throws
Exception {
    try {
      if (exceptionName.equals("First")) {
        throw new FirstException();
      } else {
        throw new SecondException();
      }
    } catch (Exception e) {
      throw e;
    }
  }
```

Rethrowing exceptions (continue)

From Java SE 7:

```
public void rethrowException(String exceptionName)
  throws FirstException, SecondException {
    try {
      // ...
    }
    catch (Exception e) {
      throw e;
    }
  }
```

Required conditions:

- The **try** block is able to throw it.
- There are no other preceding catch blocks that can handle it.
- It is a subtype or supertype of one of the catch clause's exception parameters.

1. Классификация исключений
2. Инициация и обработка исключений
3. Блок finally
4. Подводные камни
5. **Java SE 7 features**
   - 5.1 Multi-catch exceptions
   - 5.2 Rethrowing exceptions
   - 5.3 **Try-with-Resources**

Try-with-Resources

Before Java SE 7:

```
Statement stmt = null;
try {
    stmt = con.createStatement();
} catch (Exception e) {
    //do something
} finally {
  if (stmt  != null) {
        try {
            stmt .close();
        } catch (SQLException e1) {
          //do something
        }
  }
}
```

Try-with-Resources (continue)

From Java SE 7:

```
try (Statement stmt = con.createStatement()) {
    // some processing
} catch (Exception e) {
    // some handling
}
```

Try-with-Resources (continue)

```
public class MyResource implements AutoCloseable {
    @Override
    public void close() throws Exception {
        System.out.println("Closed MyResource");
    }
}
```

Try-with-Resources: AutoCloseable

Lion.java

```
package com.egar.exceptionhandling;
public class Lion implements AutoCloseable {
    public Lion() {
        System.out.println("LION is OPEN in the wild.");
    };
    public void hunt() throws Exception {
        throw new Exception("Deer Not Found!");
    }
    public void close() throws Exception {
        System.out.println("LION is CLOSED in the cage.");
        throw new Exception("Unable to close the cage!");
    }
}
```

Try-with-Resources: Example

```
public class TryWithResourcesExample {
    public static void main(String[] args) {
        try (Lion lion = new Lion()) {
            lion.hunt();
        } catch (Exception e) {
            System.out.println(e);
        } finally {
            System.out.println("Finally.");
        }
    }

}
```

```
LION is OPEN in the wild.
LION is CLOSED in the cage.
java.lang.Exception: Deer Not Found!
Finally.
```