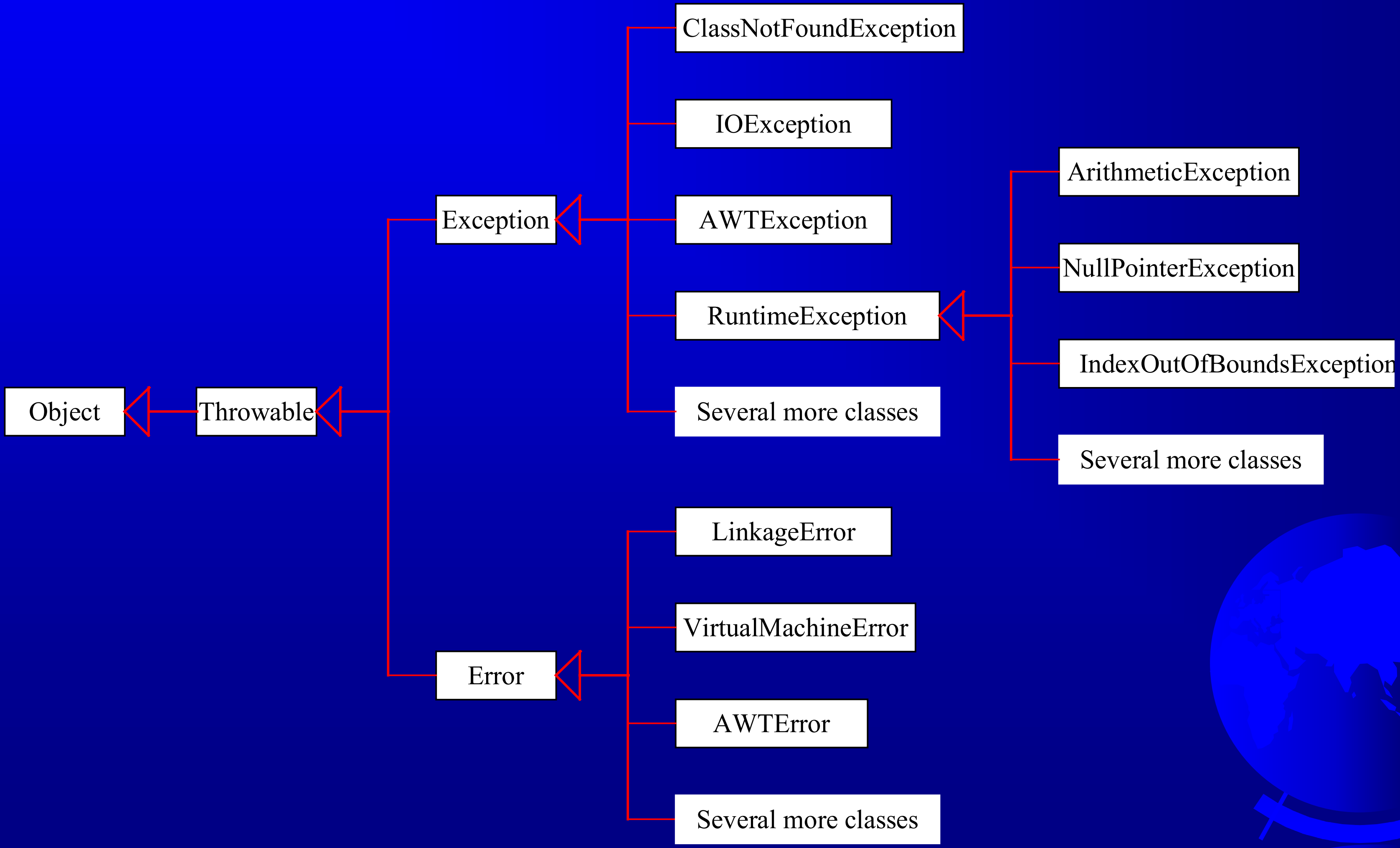
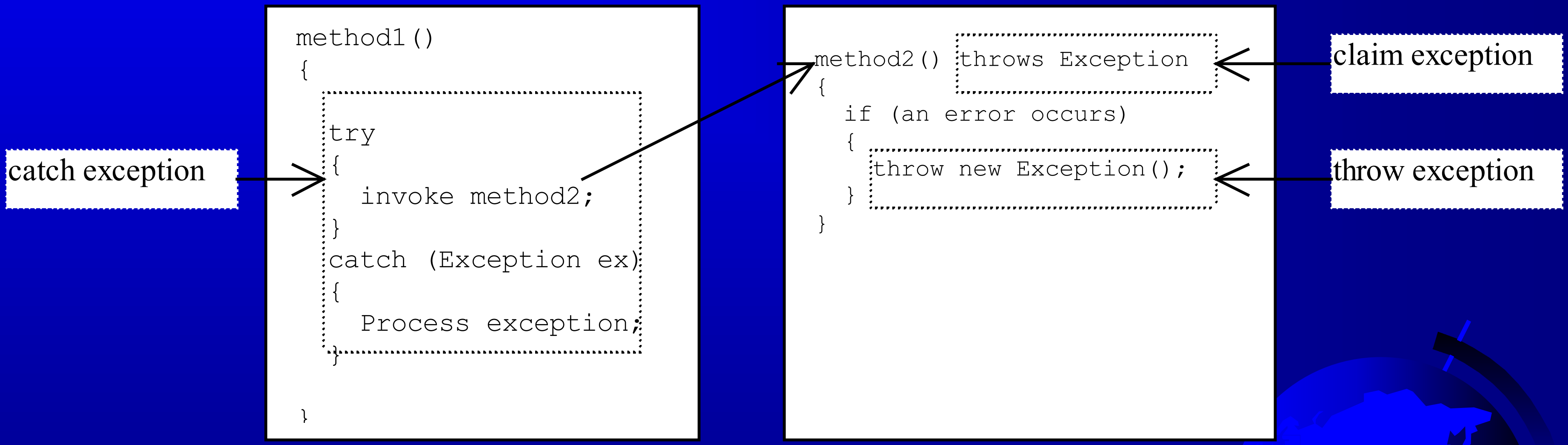
Chapter 11: Exception Handling

- Exceptions and Exception Types
- Claiming Exceptions
- Throwing Exceptions
- Catching Exceptions
- Rethrowing Exceptions
- The `finally` Clause
- Cautions When Using Exceptions
- Creating Your Own Exception Classes (Optional)

Exceptions and Exception Types



Claiming, Throwing, and Catching Exceptions



```
method1()
{

  try
  {
    invoke method2;
  }
  catch (Exception ex)
  {
    Process exception;
  }

}
```

```
method2() throws Exception
{
  if (an error occurs)
  {
    throw new Exception();
  }
}
```

Claiming Exceptions

-
  ```
  public void myMethod()
    throws IOException
  ```

-
  ```
  public void myMethod()
    throws IOException, OtherException
  ```

Throwing Exceptions

- ☞ `throw new TheException();`
- ☞
  ```
  TheException e = new TheException();
  throw e;
  ```

Throwing Exceptions Example

```
public Rational divide(Rational r) throws
  Exception
{
  if (r.numer == 0)
  {
    throw new Exception("denominator
      cannot be zero");
  }

  long n = numer*r.denom;
  long d = denom*r.numer;
  return new Rational(n,d);
}
```

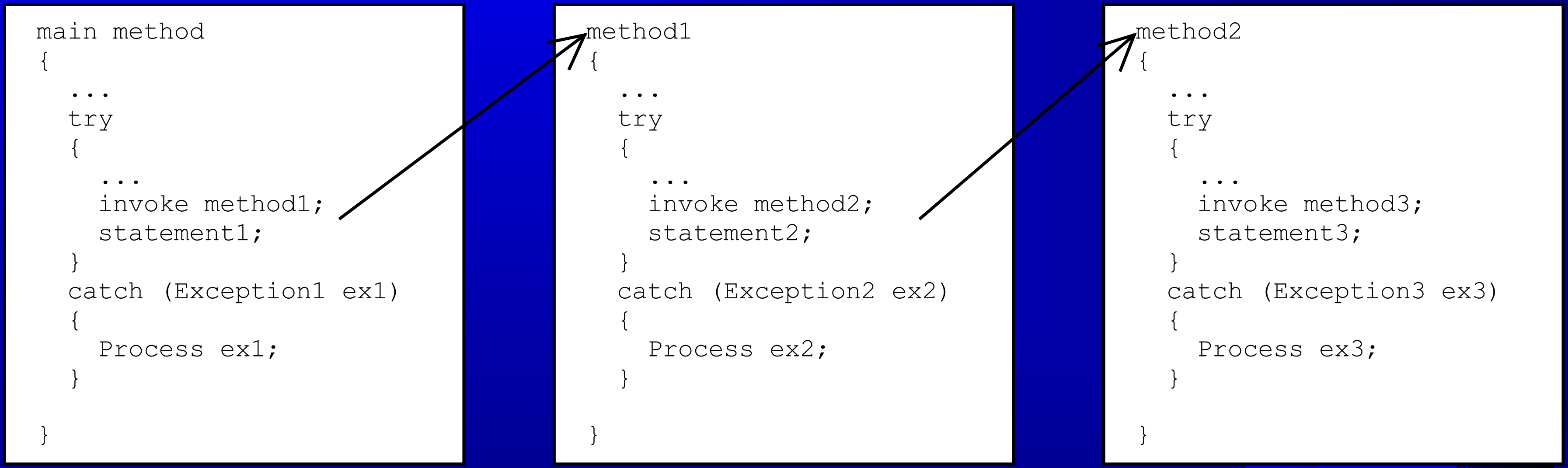
Catching Exceptions

```
try
{
  statements;
}
catch (Exception1 e)
{ handler for exception1 }
catch (Exception2 e)
{ handler for exception2 }

...

catch (ExceptionN e)
{   handler for exceptionN   }
```

Catching Exceptions



```
main method
{
  ...
  try
  {
    ...
    invoke method1;
    statement1;
  }
  catch (Exception1 ex1)
  {
    Process ex1;
  }

}
```

```
method1
{
  ...
  try
  {
    ...
    invoke method2;
    statement2;
  }
  catch (Exception2 ex2)
  {
    Process ex2;
  }

}
```

```
method2
{
  ...
  try
  {
    ...
    invoke method3;
    statement3;
  }
  catch (Exception3 ex3)
  {
    Process ex3;
  }

}
```

Example 11.1 Claiming, Throwing, and Catching Exceptions

- Objective: Write a program to test the new `Rational` class.

TestRationalException

Rational

Run

Example 11.2 Exceptions in GUI Applications

- An error message appears on the console, but the GUI application continues running.
- Re-run the MenuDemo applet from Example 9.9 and divide by 0 to see how a GUI deals with unhandled exceptions.

MenuDemo

Run

Rethrowing Exceptions

```
try
{
  statements;
}
catch(TheException e)
{
  perform operations before exits;
  throw e;
}
```

The `finally` Clause

```
try
{
  statements;
}
catch(TheException e)
{
  handling e;
}
finally
{
  finalStatements;
}
```

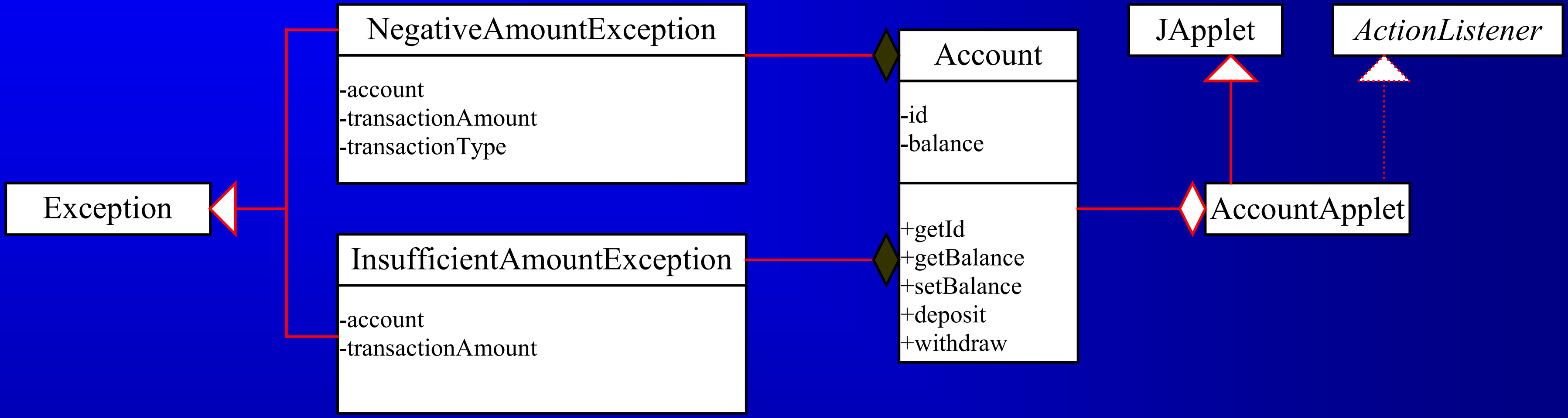
Cautions When Using Exceptions

- Exception handling separates error-handling code from normal programming tasks, thus making programs easier to read and to modify. Be aware, however, that exception handling usually requires more time and resources because it requires instantiating a new exception object, rolling back the call stack, and propagating the errors to the calling methods.

Example 11.3 (Optional) Creating Your Own Exception Classes

- Objective: This program creates a Java applet for handling account transactions. The applet displays the account id and balance, and lets the user deposit to or withdraw from the account. For each transaction, a message is displayed to indicate the status of the transaction: successful or failed. In case of failure, the failure reason is reported.

Example 11.3, cont.



InsufficientFundException

Account

NegativeAmountException

Run