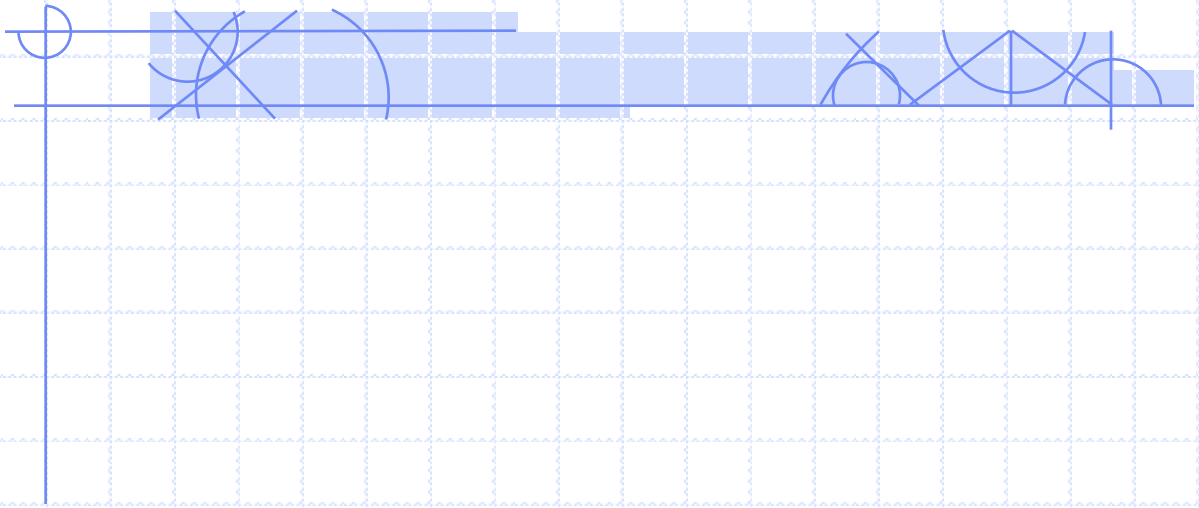
Exceptions



Definition

- An ***exception*** represents an error condition that can occur during the normal execution of the program.

- When an exception occurs, or is ***thrown***, the normal sequence of flow is terminated. The exception-handling routine is then executed; we say the thrown exception is ***caught***.

Use of exception

- Exception
  - Indication of problem during execution
    - E.g., divide by zero
- Uses of exception handling
  - Process exceptions from program components
  - Handle exceptions in a uniform manner in large projects
  - Remove error-handling code from “main line” of execution

Exception Throwers

- An ***exception thrower*** is a method that throws an exception. This exception may be created by the method it self (***Exception Generator***), or thrown by another called method (***Exception Propagator***).
- The `throws` clause specifies exceptions thrown by the method:
  - public int divide(int a, int b) throws `ArithmeticException`
  - public boolean contains(T o) throws `NullPointerException`

Exception Generators

```
public boolean contains(T o) throws NullPointerException {
boolean found = false;
if (o == null) throw new NullPointerException();
…
return found;
}
```

```
public int divide(int a, int b) throws ArithmeticException {
int r;
if ( b == 0) throw new ArithmeticException() ;
r = a / b;
return r;
}
```

```
public boolean contains(T o, int pos) throws Exception {
boolean found = false;
if (o == null) throw new NullPointerException();
if (pos < 0 || pos > count) throw new BoundaryException();
…
return found;
}
```

Simple Exception Propagator

```
public … propagator(int a, int b) throws Exception {
…
generator(…);
…
}
```

```
public … generator(…) throws Exception {
…
if (ErrorCondition) throw new Exception();
…

}
```

Simple Exception Propagator

```
public … propagator(int a, int b) throws Exception {
…
thrower(…);
…
}
```

```
public … thrower(…) throws Exception {
…
…

}
```

Exception Catcher

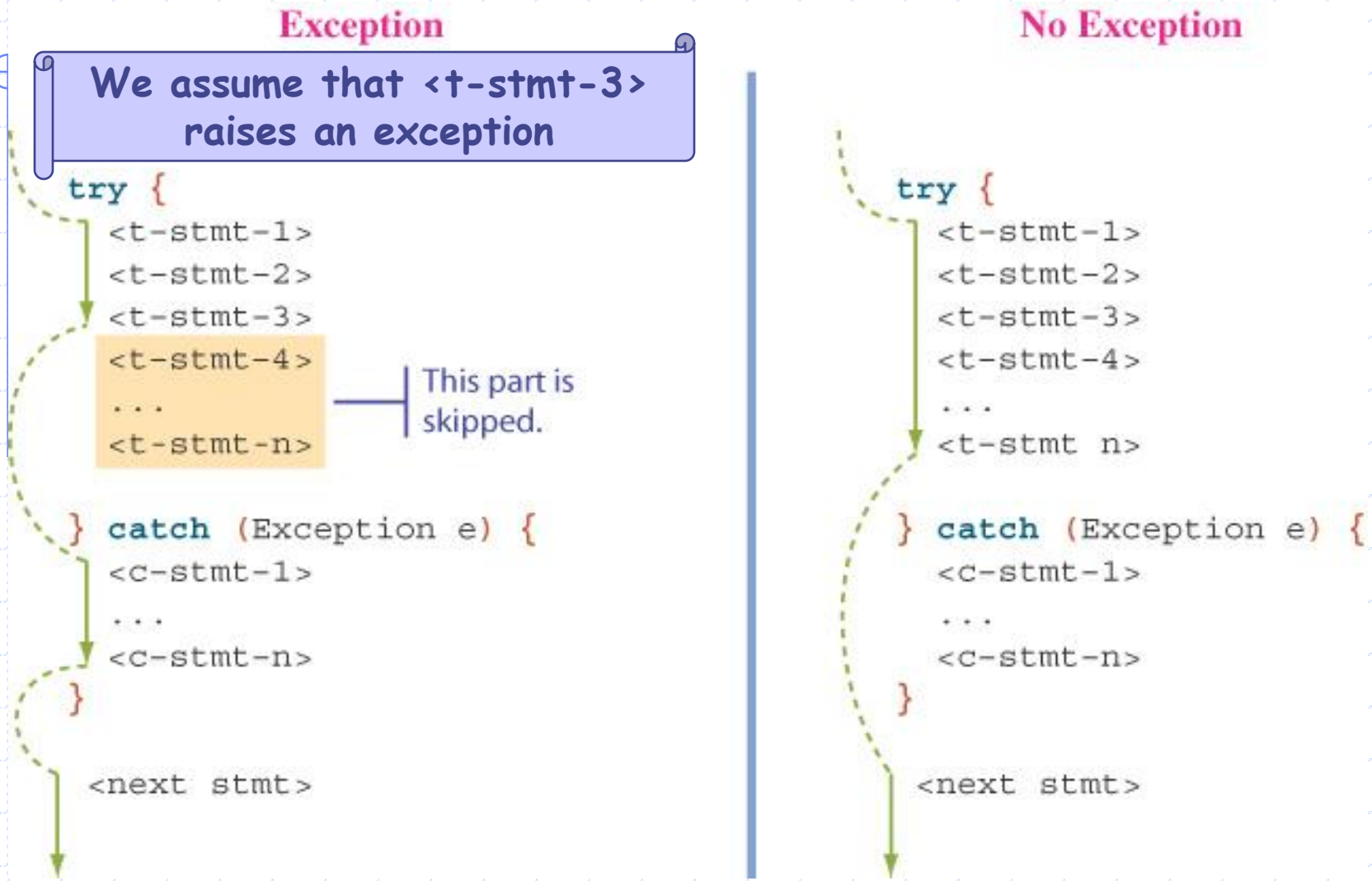
- An ***exception catcher*** is any method that includes a matching **catch** block for the thrown exception.
- The catcher uses the clause try and catch.
  - Code that could generate errors put in `try` blocks
  - Code for error handling enclosed in a `catch` clause
- A method may be a catcher of one exception and a propagator of another (It is also a propagator).

Simple Exception Propagator

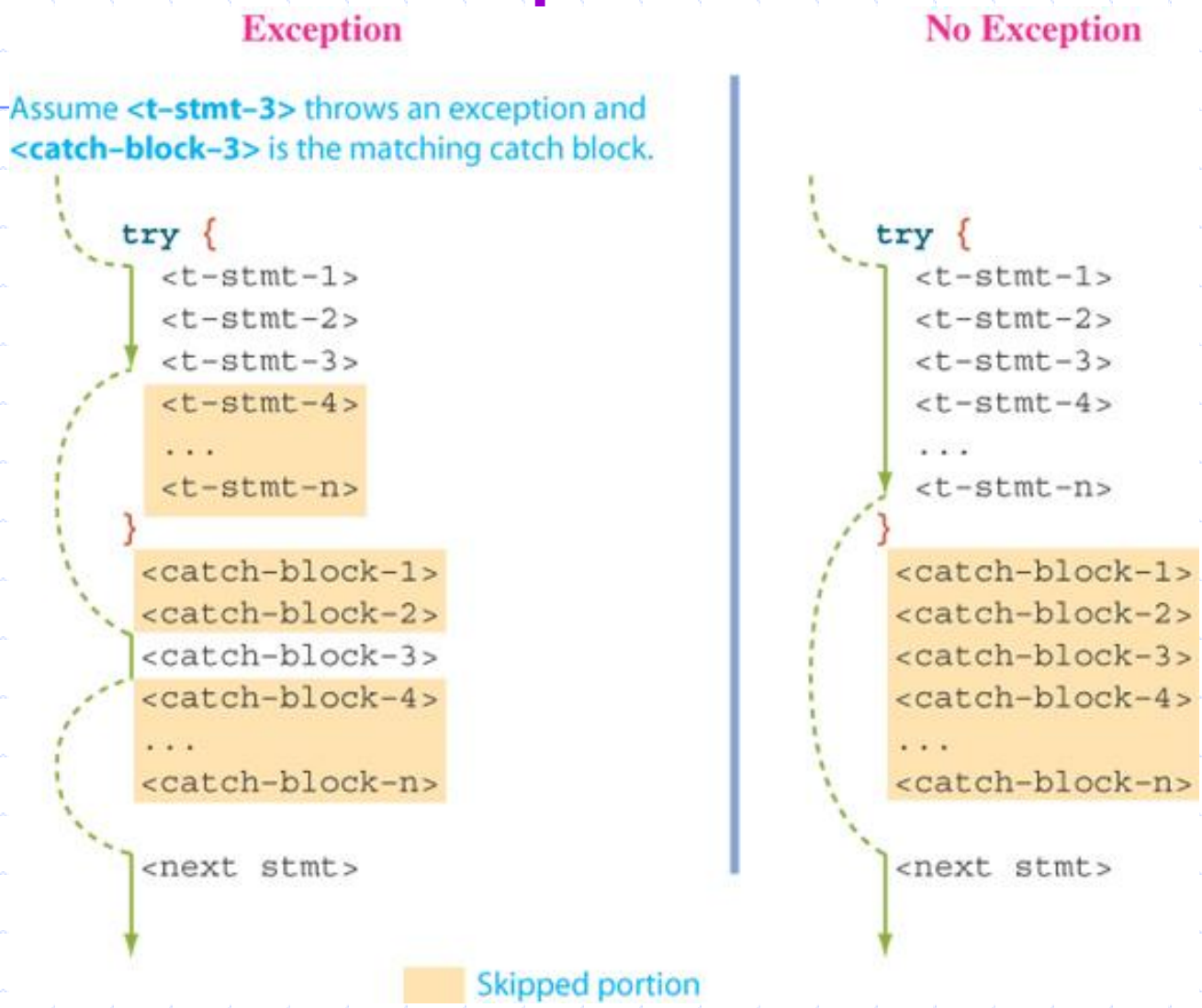
```
public … catcher(int a, int b) {
…
try {

        thrower(…);

    } catch ( Exception e) { ….. }
…

}
```

```
public … thrower(…) throws Exception {
…

}
```

try-catch Control Flow



Multiple catch



Thrower's Caller cases

- A thrower may be called by :
  - An other Thrower:
    - The caller throws the same exception
      - Without any additional processing : the caller is a simple propagator: Case 1
      - With additional processing: the caller is a catcher-propagator: Case 2
    - The caller throws an other exception: the caller is a catcher-propagator : Case 3
  - An non thrower: the caller is a catcher: Case 4.

Case 1 Example

```
public … f(…) throws Exception {
…
g(…);
…
}
```

```
public … g(…) throws Exception {
…
if (ErrorCondition) throw new Exception();
…

}
```

Case 2 Example

```
public … f(…) throws Exception {
…
try {
g(…);
} catch(Exception ex) {
…
throw ex;
}
…
}
```

```
public … g(…) throws Exception {
…
if (ErrorCondition) throw new Exception();
…

}
```

Case 3 Example

```
public … f(int a, int b) throws OtherException {
…
try {
g(…);
} catch(Exception ex) {
…
throw new OtherException(…);
}
…
}
```

```
public … g(…) throws Exception {
…
if (ErrorCondition) throw new Exception();
…

}
```

Case 4 Example

```
public … f(int a, int b) {
…
try {
g(…);
} catch(Exception ex) {
…
}
…
}
```

```
public … g(…) throws Exception {
…
if (ErrorCondition) throw new Exception();
…

}
```

Thrower with Multiple Exceptions

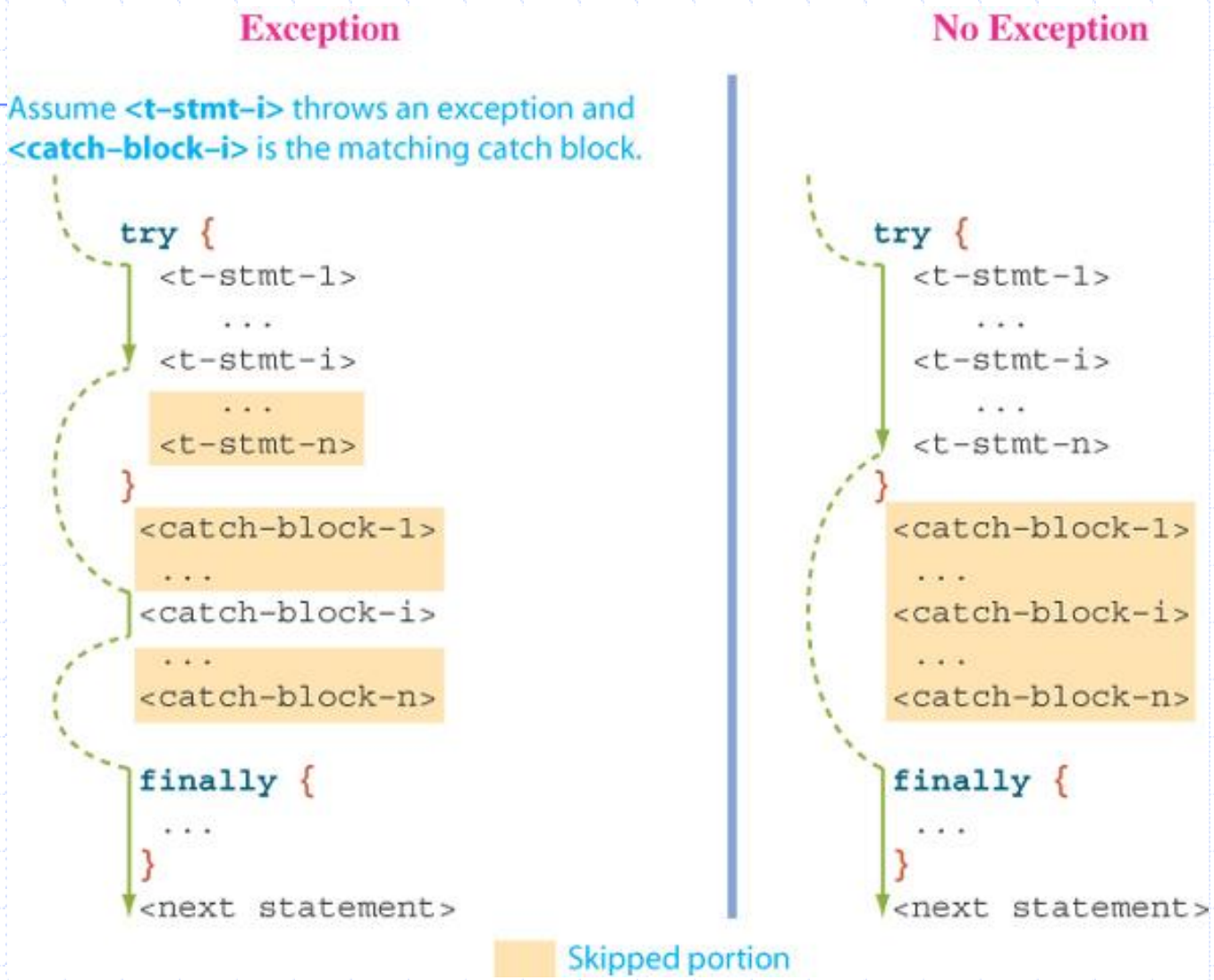
```
public … f(…) {
…
try {
contains( p, 5);
}
catch(NullPointerException ex) {
…
}
catch(BoudaryException ex) {
…
}

…
}
```

```
public boolean contains(T o, int pos) throws Exception {
boolean found = false;
if (o == null) throw new NullPointerException();
if (pos < 0 || pos > count) throw new BoundaryException();
…
return found;
}
```

The finally Block

- There are situations where we need to take certain actions regardless of whether an exception is thrown or not.
- We place statements that must be executed regardless of exceptions in the finally block.

try-catch-finally Control Flow



Exception Types

- All types of thrown errors are instances of the **Throwable** class or its subclasses.
- Serious errors are represented by instances of the **Error** class or its subclasses.
- Exceptional cases that common applications should handle are represented by instances of the **Exception** class or its subclasses.

Throwable Hierarchy

- There are over 60 classes in the hierarchy.