Programming Languages

2nd edition

Tucker and Noonan

Chapter 7

Semantics

Ismael: "Surely all this is not without meaning."

Herman Melville, Moby Dick

Contents

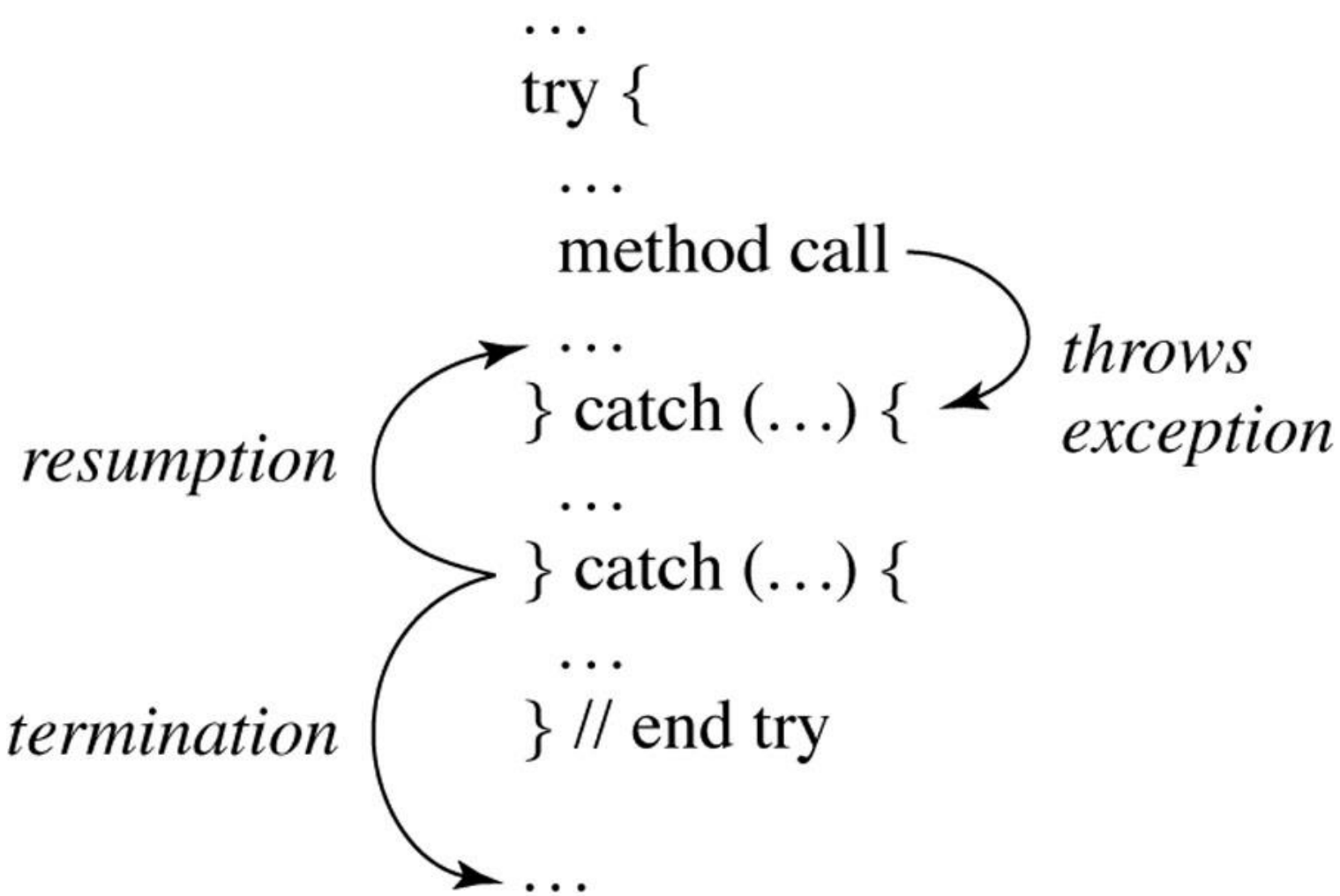
Purpose

- Simplify programming
- Make applications more *robust*.
- What does *robust* mean?

```
(* Pascal - what can go wrong *)
reset(file, name);
  (* open *)
sum := 0.0;

count := 0;

while (not eof(file)) do begin

   read(file, number);
```

Exception Handling

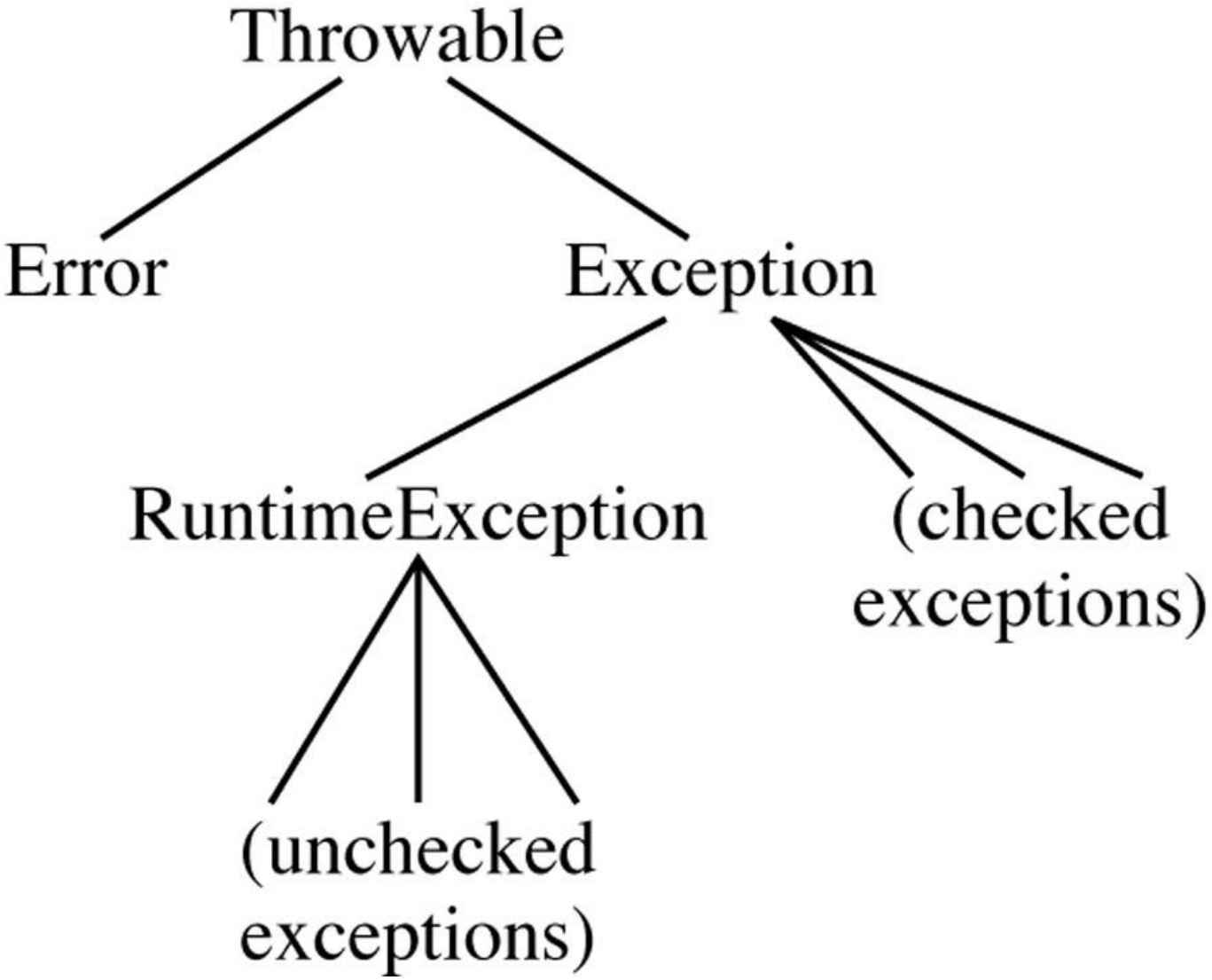
Figure 7.9



```
#include <iostream.h>
int main () {
  char A[10];
  cin >> n;
  try {
    for (int i=0; i<n; i++){
      if (i>9) throw "array index error";
      A[i]=getchar();
    }
  }
  catch (char* s)
  { cout << "Exception: " << s << endl; }
  return 0;
}
```

Java Exception Type Hierarchy

Figure 7.10



Creating a New Exception Class

```
class StackUnderflowException extends Exception {
   public StackUnderflowException() { super(); }
   public StackUnderflowException(String s){  super(s);}
}
```

Missing Argument Exception

```
public static void main(String[] arg) {
    try {
    if (arg.length < 1)  {
    System.err.println("Missing argument.");
    displayUsage( );
    System.exit(1);
    }
    process(new BufferedReader(new FileReader(arg[0])));
  } catch (FileNotFoundException e) {
    System.err.println("Cannot open file: " + arg[0]);
    System.exit(1);
  }
}
```

Invalid Input Exception

Figure 7.12

```
while (true) {
  try {
    System.out.print ( "Enter number : ");
    number = Integer.parseInt(in.readLine ( ));
    break;
  } catch (NumberFormatException e) {
    System.out.println ( "Invalid number, please reenter.");
  } catch (IOException e) {
    System.out.println("Input error occurred, please reenter.");
  } // try
} // while
```

StackUnderflowException Class

Figure 7.13

```
class StackUnderflowException extends Exception {
  public StackUnderflowException( ) { super(); }
  public StackUnderflowException(String s){ super(s);}
}
```

Throwing an Exception

Figure 7.13

```
class Stack {
  int stack[];
  int top = 0;
  ...

  public int pop() throws StackUnderflowException {
    if (top <= 0)
        throw new StackUnderflowException("pop on empty stack");
    return stack[--top];
  }
  ...
}
```

AssertException Class

```
class AssertException extends RuntimeException {
    public AssertException( ) { super( ); }

    public AssertException(String s) { super(s); }
}
```

Assert Class

Figure 7.15

```
class Assert {
   static public final boolean ON = true;
   static public void isTrue(boolean b) {
      if (!b) { throw new AssertException("Assertion failed"); }
   }

   static public void shouldNeverReachHere() {
      throw new AssertException("Should never reach here");
   }
}
```

Using Asserts

```
class Stack {
  int stack[];
  int top = 0;
  ...

  public boolean empty() { return top <= 0; }

  public int pop() {
    Assert.isTrue(!empty());
    return stack[--top];
  }
  ...
}
```