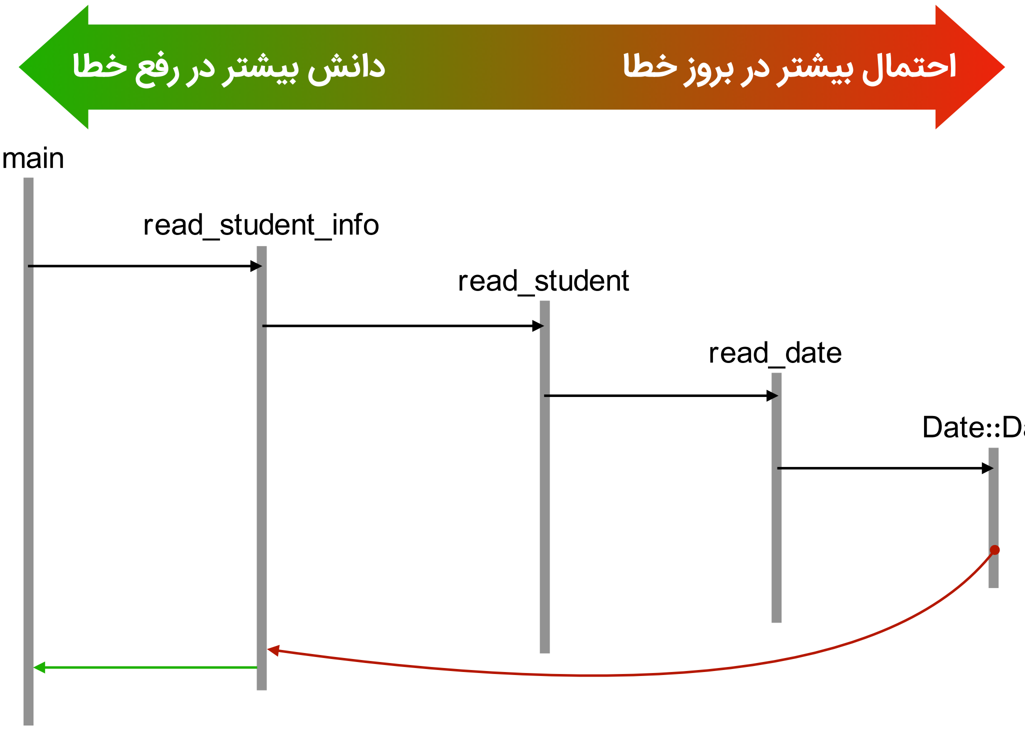
دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

رسیدگی به خطاها — ۴

رسیدگی به خطاها با «استثناءها»

برنامه‌سازی پیشرفته — رامتین خسروی

بهار ۹۹



نیاز به سازوکاری برای انتقال حالت خطا از محل بروز خطا به محل رسیدگی به خطا

به سازوکاری برای رسیدگی به خطا نیاز داریم که:

- خطا را به طور خودکار تا نقطه رسیدگی منتشر کند
- ما را مجبور به اضافه‌کردن کدهای بی‌مورد نکند

read_student_info

```
void read_student_info(char* filename,
        vector<Student>& v)
{
  ifstream input(filename);
  int count;
  input >> count;
  for (int i = 0; i < count; i++) {
    try {
      Student s = read_student(input);
      v.push_back(s);
    } catch(runtime_error& ex) {
      input.clear();
      string to_be_ignored;
      getline(input, to_be_ignored);
    }
  }
  input.close();
}
```

read_student

```
Student read_student(ifstream& input)
{
  string name;
  input >> name;
  Date bdate = read_date(input);
  return Student(name, bdate);
}
```

read_date

```
Date read_date(ifstream& input)
{
  int d, m, y;
  char ch;
  input >> d;
  input >> ch;
  if (ch != '/')
    throw runtime_error("…");
  input >> m;
  input >> ch;
  if (ch != '/')
    throw runtime_error("…");
  input >> y;
  return Date(d, m, y);
}
```

https://github.com/ramtung/apnotes/tree/master/10_Exceptions/Import.3-ErrorHandlingByException

read_student_info

```
void read_student_info(char* filename, vector<Student>& v) {
    ifstream input(filename);
    int count;
    input >> count;
    for (int i = 0; i < count; i++) {
        try {
            Student s = read_student(input);
            v.push_back(s);
        } catch(runtime_error& ex) {
            input.clear();
            string to_be_ignored;
            getline(input, to_be_ignored);
        }
    }
    input.close();
}
```

read_student_info

```
void read_student_info(char* filename,
        vector<Student>& v)
{
  ifstream input(filename);
  int count;
  input >> count;
  for (int i = 0; i < count; i++) {
    try {
      Student s = read
      v.push_back(s);
    } catch(runtime_er
      input.clear();
      string to_be_ign
      getline(input, t
    }
  }
  input.close();
}
```

read_student

```
Student read_student(ifstream& input) {
    string name;
    input >> name;
    Date bdate = read_date(input);
    return Student(name, bdate);
}
```

ate

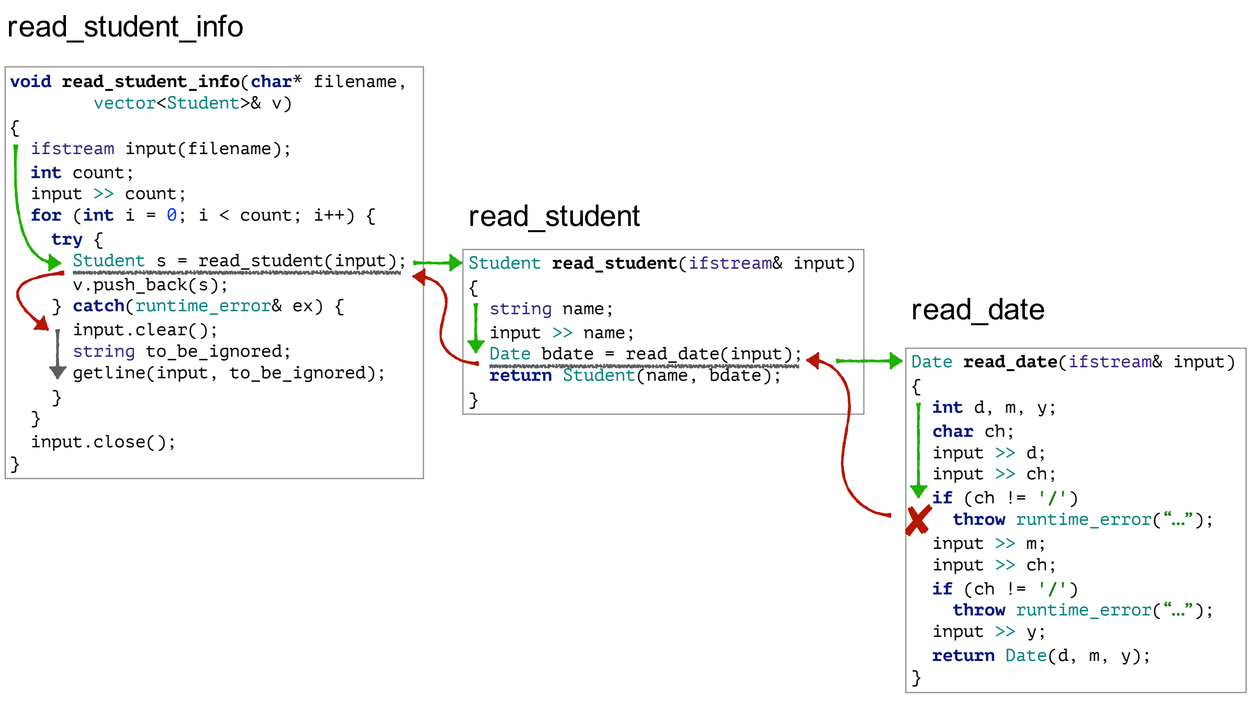
```
_date(ifstream& input)
n, y;
;
> d;
> ch;
!= '/')
  throw runtime_error("…");
input >> m;
input >> ch;
if (ch != '/')
  throw runtime_error("…");
input >> y;
return Date(d, m, y);
}
```

read_student_info

```
void read_student_info(char* filename,
        v
{
  ifstrea
  int cou
  input >
  for (in
    try {
      Stu
      v.p
    } cat
      inp
      str
      get
    }
  }
  input.c
}
```

```
Date read_date(ifstream& input) {
    int d, m, y;
    char ch;
    input >> d;
    input >> ch;
    if (ch != '/')
        throw runtime_error("Slash separator expected");
    input >> m;
    input >> ch;
    if (ch != '/')
        throw runtime_error("Slash separator expected");
    input >> y;
    return Date(d, m, y);
}
```

```
input)
…");
…");
}
```



read_student_info

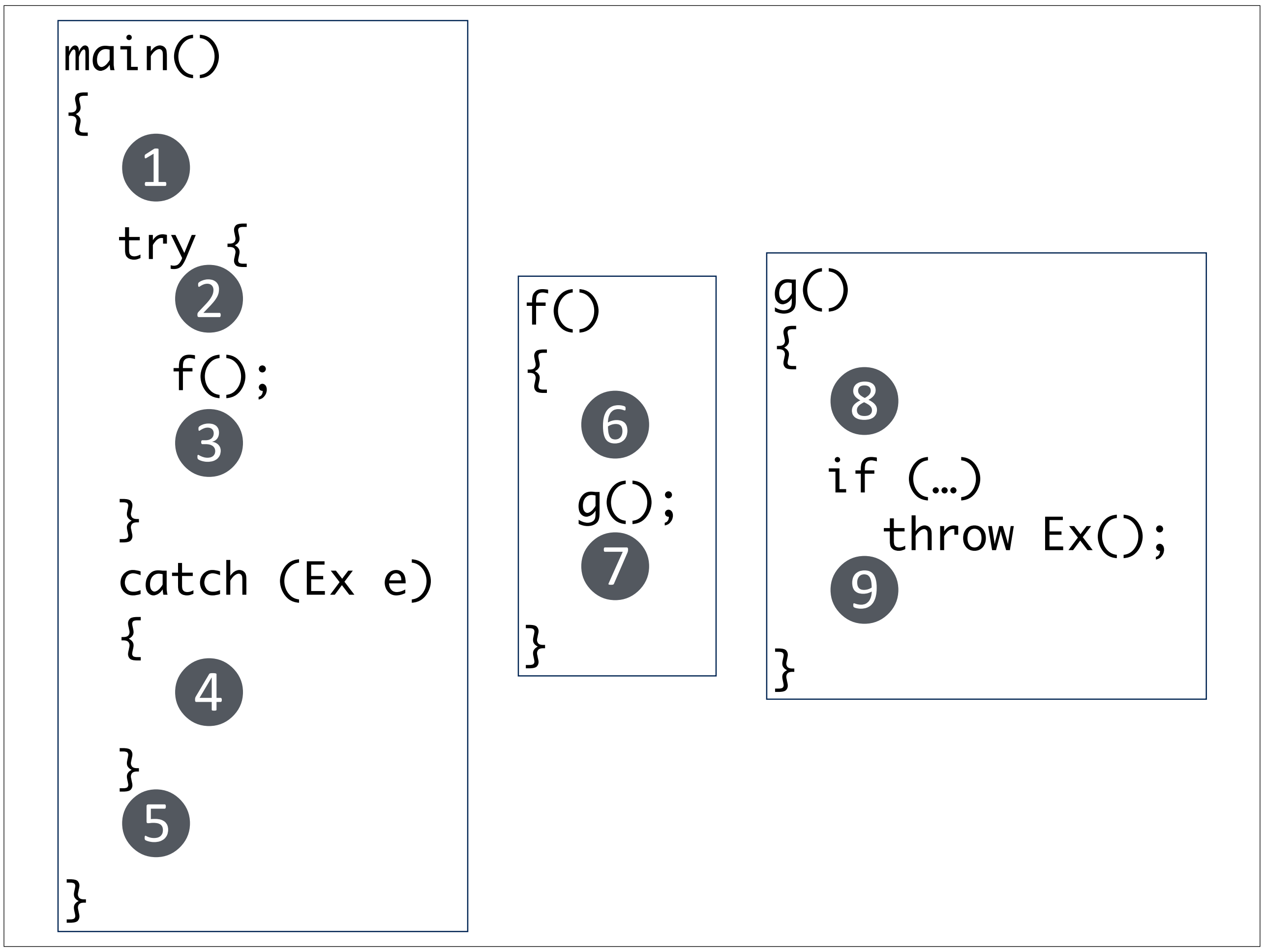
```
void read_student_info(char* filename,
        vector<Student>& v)
{
  ifstream input(filename);
  int count;
  input >> count;
  for (int i = 0; i < count; i++) {
    try {
      Student s = read_student(input);
      v.push_back(s);
    } catch(runtime_error& ex) {
      input.clear();
      string to_be_ignored;
      getline(input, to_be_ignored);
    }
  }
  input.close();
}
```

read_student

```
Student read_student(ifstream& input)
{
  string name;
  input >> name;
  Date bdate = read_date(input);
  return Student(name, bdate);
}
```

read_date

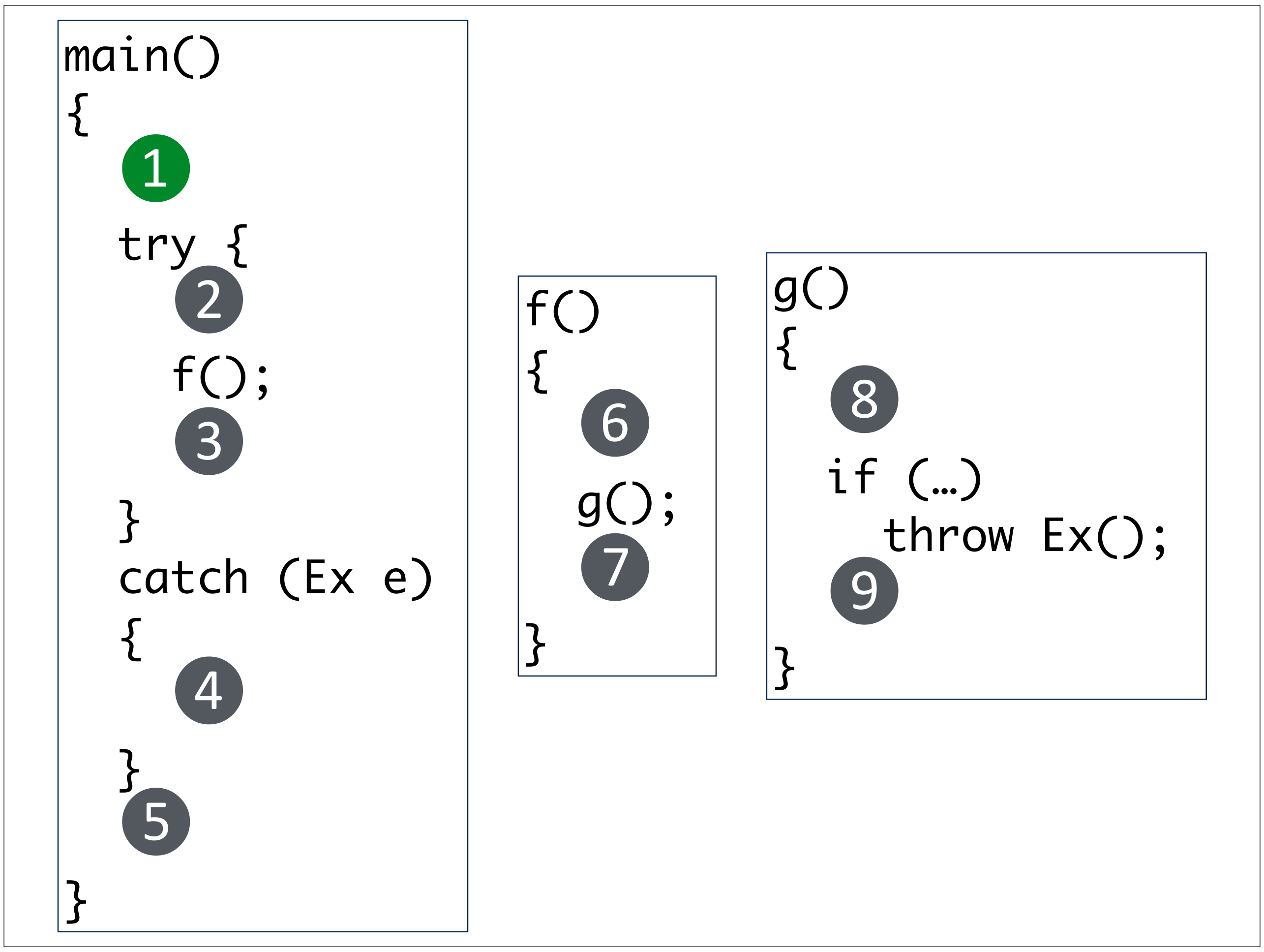
```
Date read_date(ifstream& input)
{
  int d, m, y;
  char ch;
  input >> d;
  input >> ch;
  if (ch != '/')
    throw runtime_error("…");
  input >> m;
  input >> ch;
  if (ch != '/')
    throw runtime_error("…");
  input >> y;
  return Date(d, m, y);
}
```



```
main()
{
  1
  try {
    2
    f();
    3
  }
  catch (Ex e)
  {
    4
  }
  5
}
```

```
f()
{
    6
    g();
    7
}
```

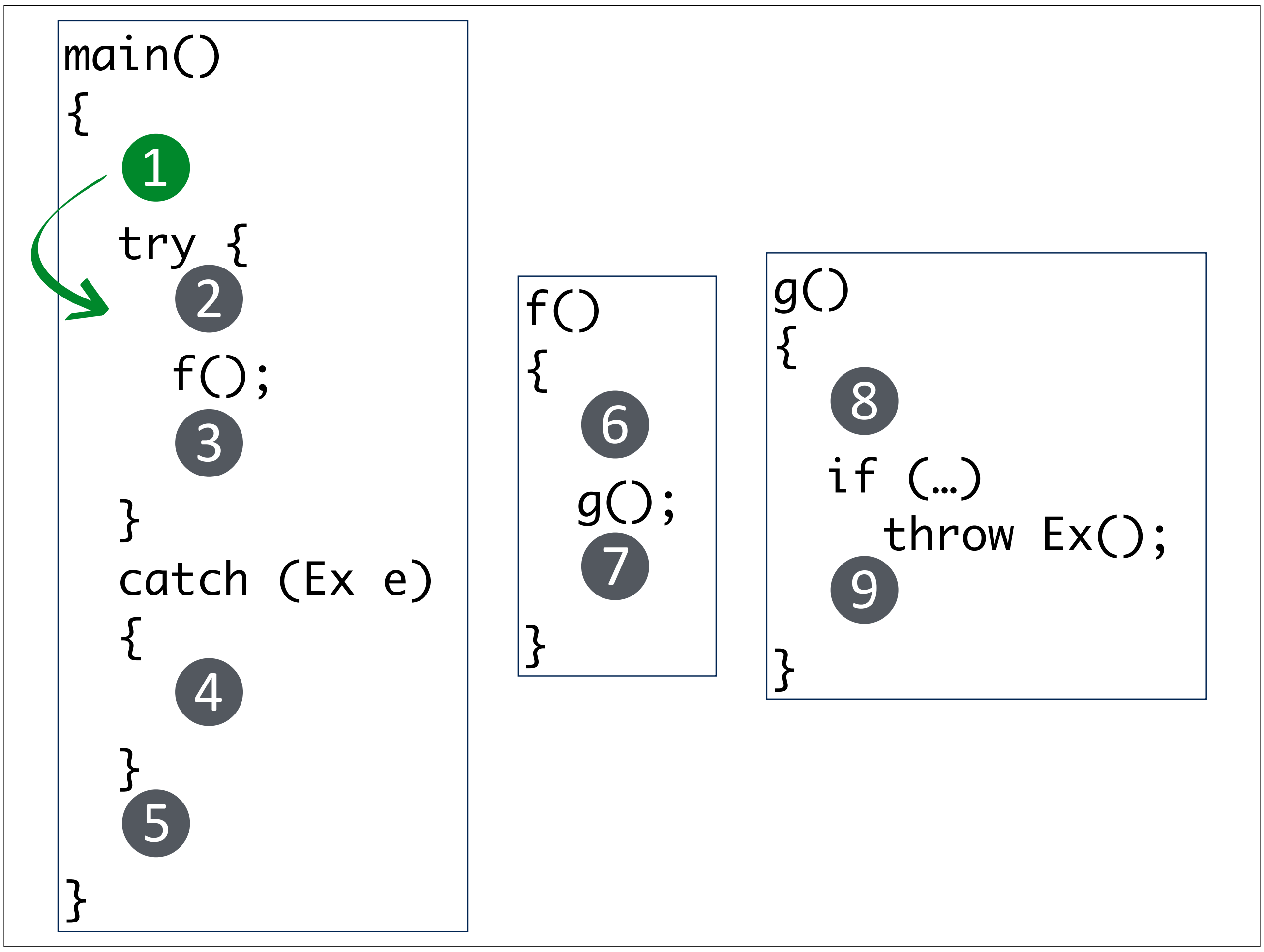
```
g()
{
  8
  if (…)
    throw Ex();
  9
}
```



```
main()
{
  1
  try {
    2
    f();
    3
  }
  catch (Ex e)
  {
    4
  }
  5
}
```

```
f()
{
    6
    g();
    7
}
```

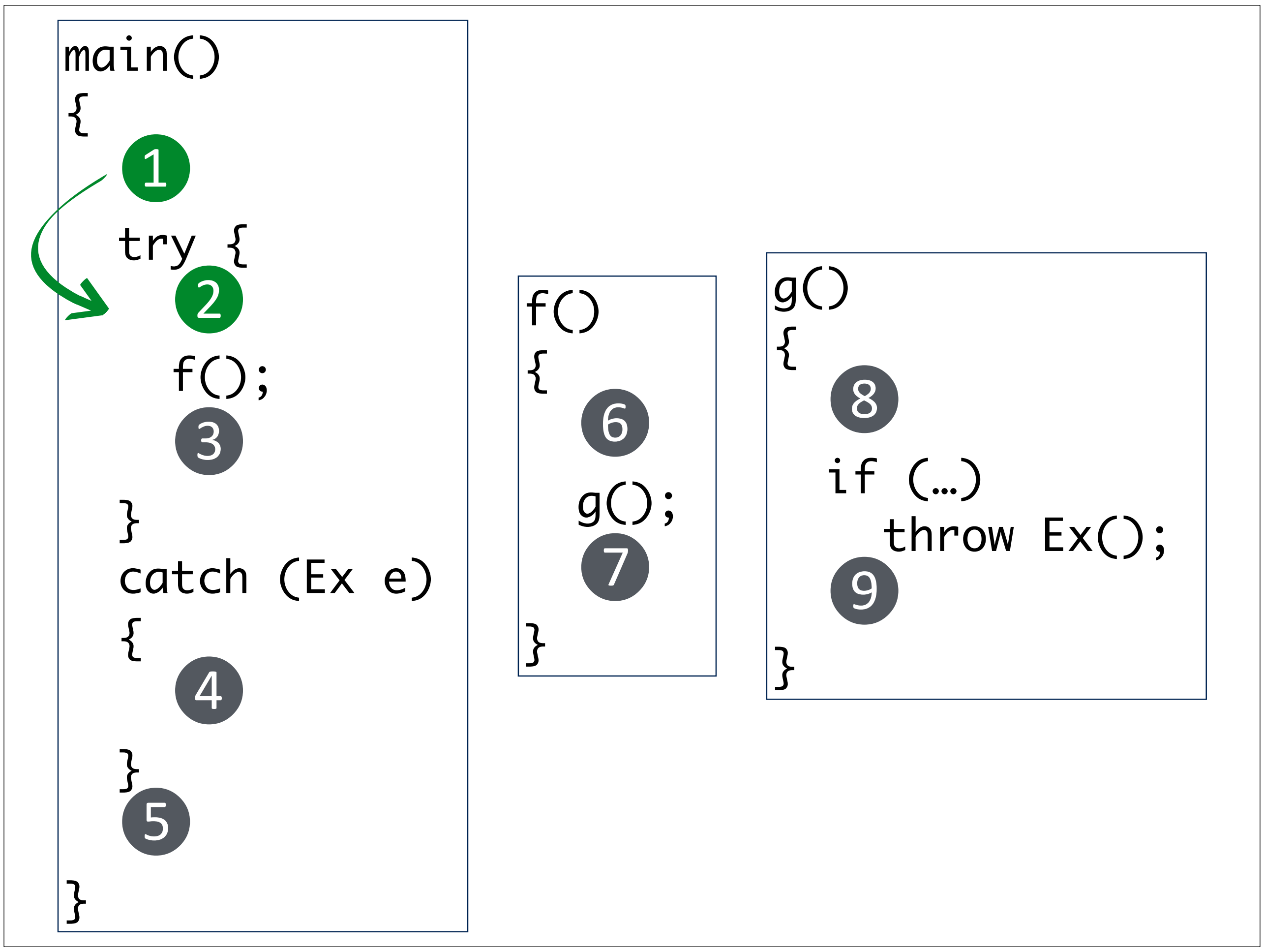
```
g()
{
  8
  if (…)
    throw Ex();
  9
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

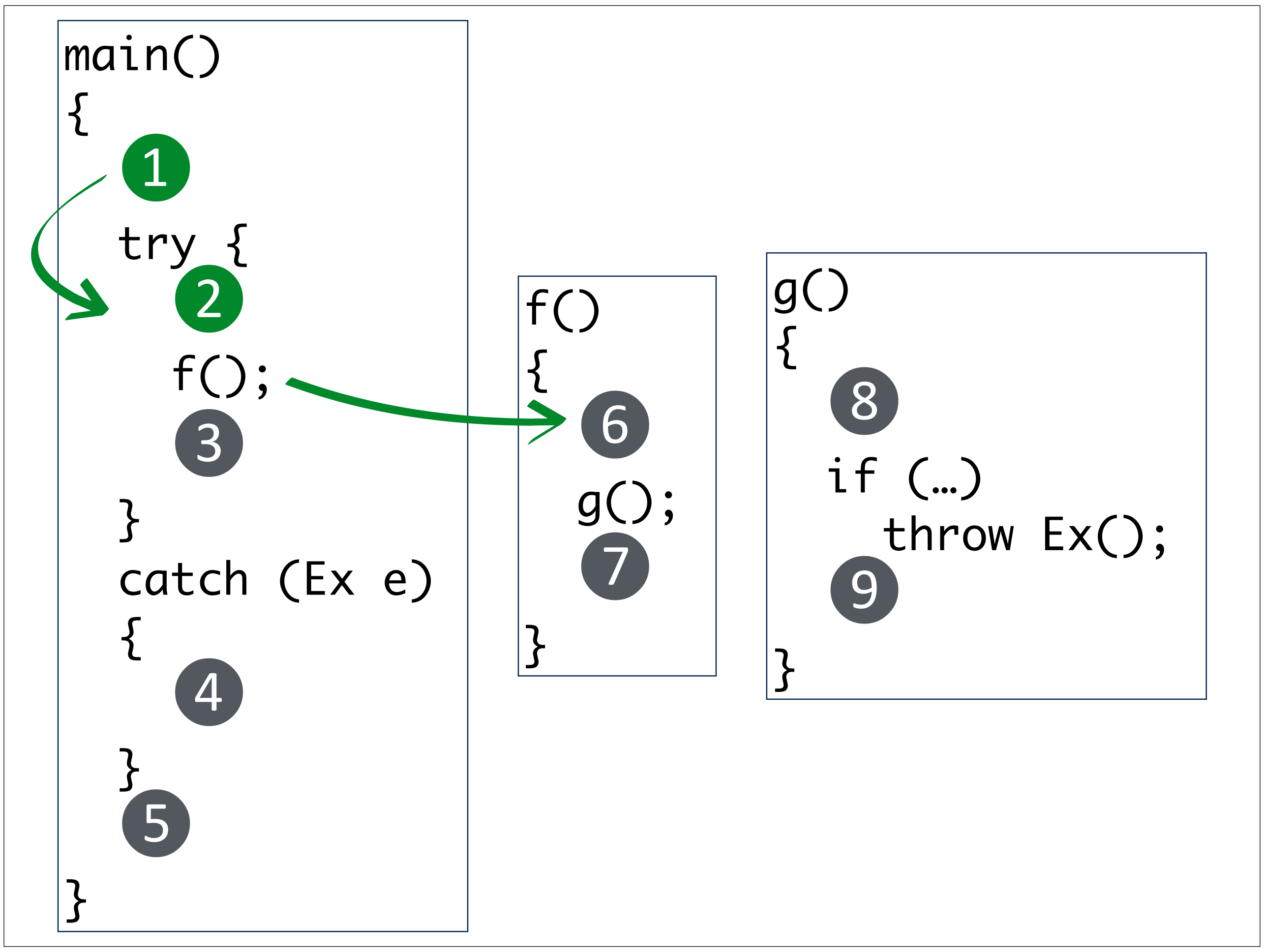
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

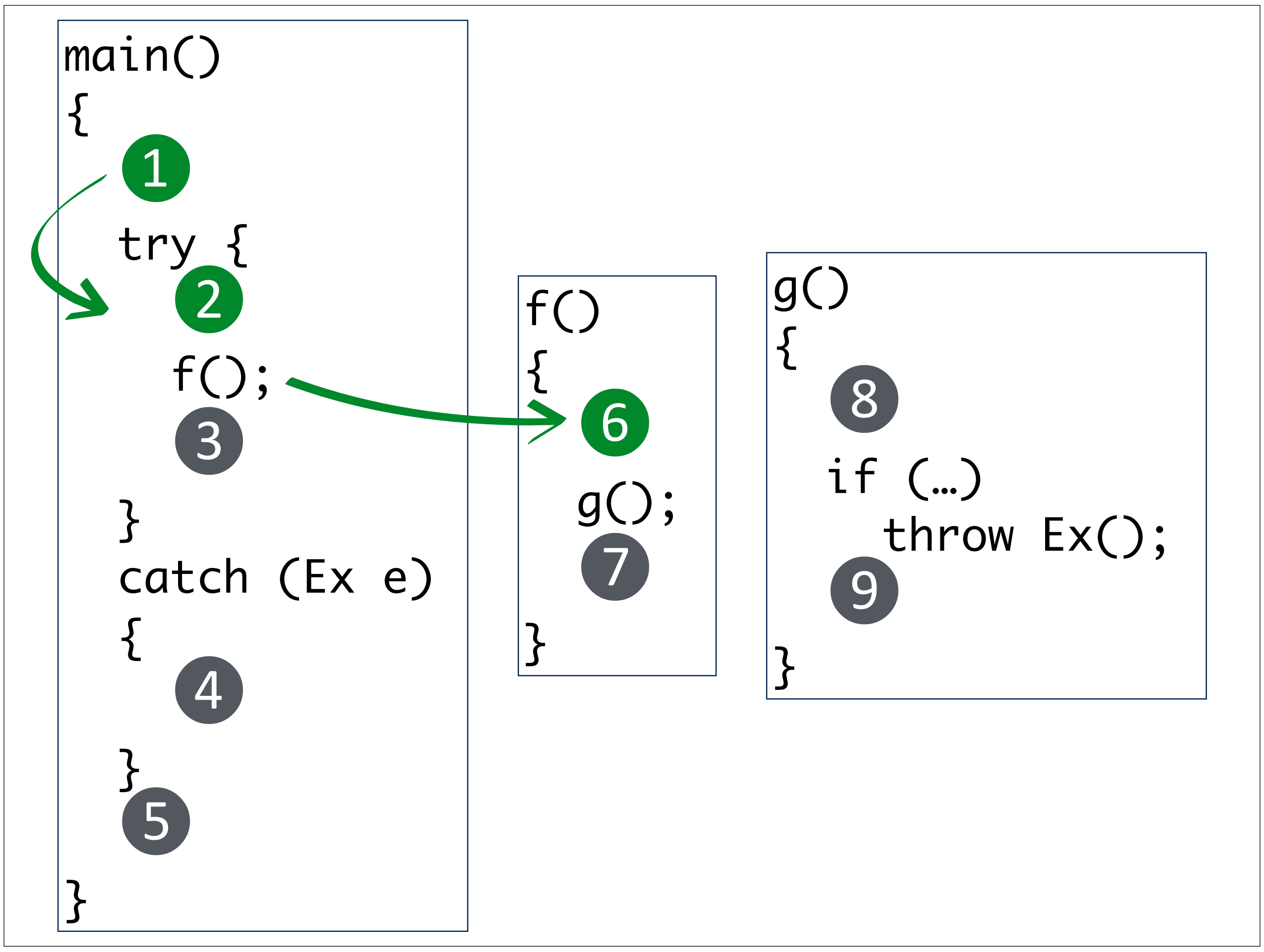
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

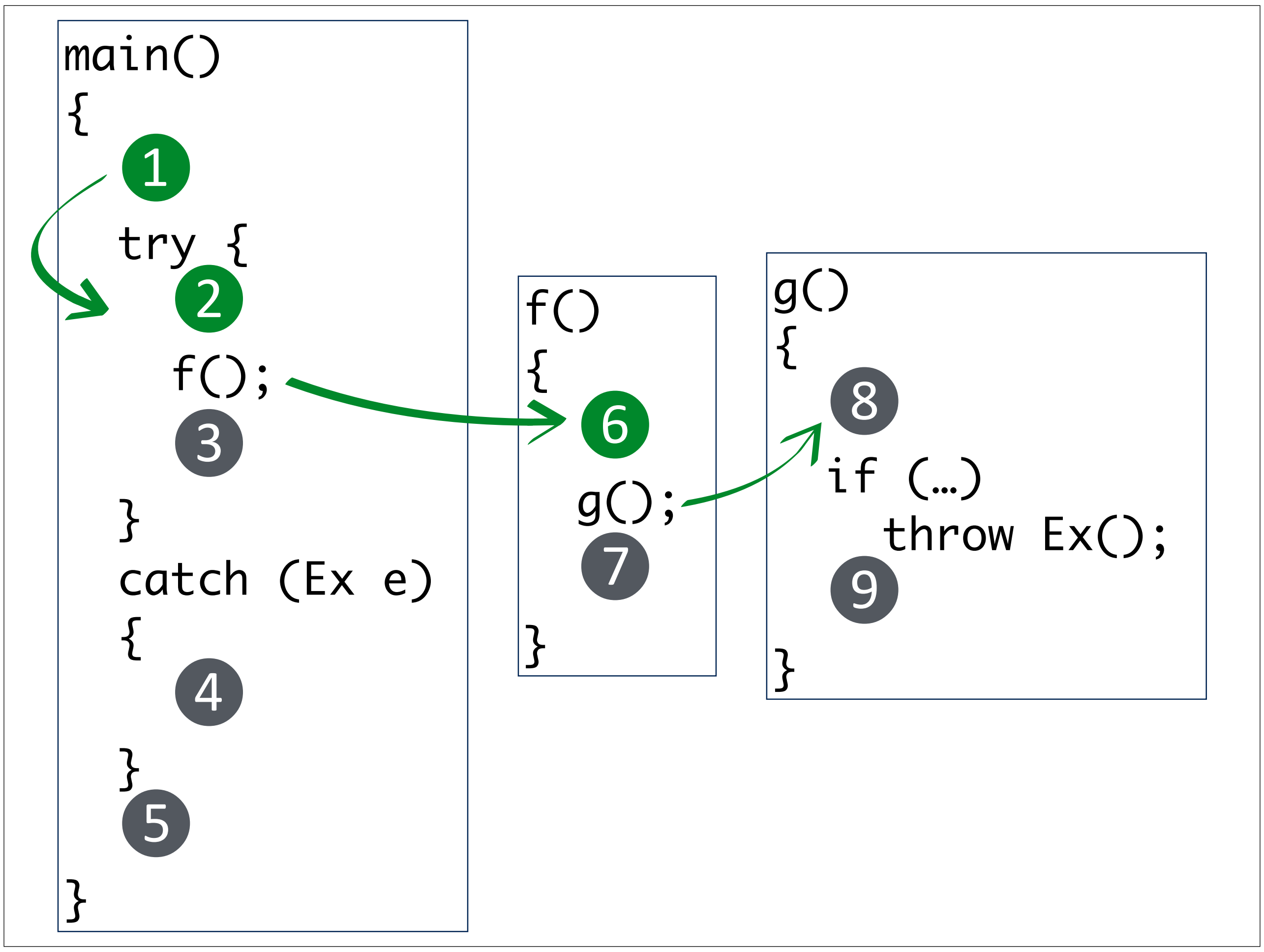
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

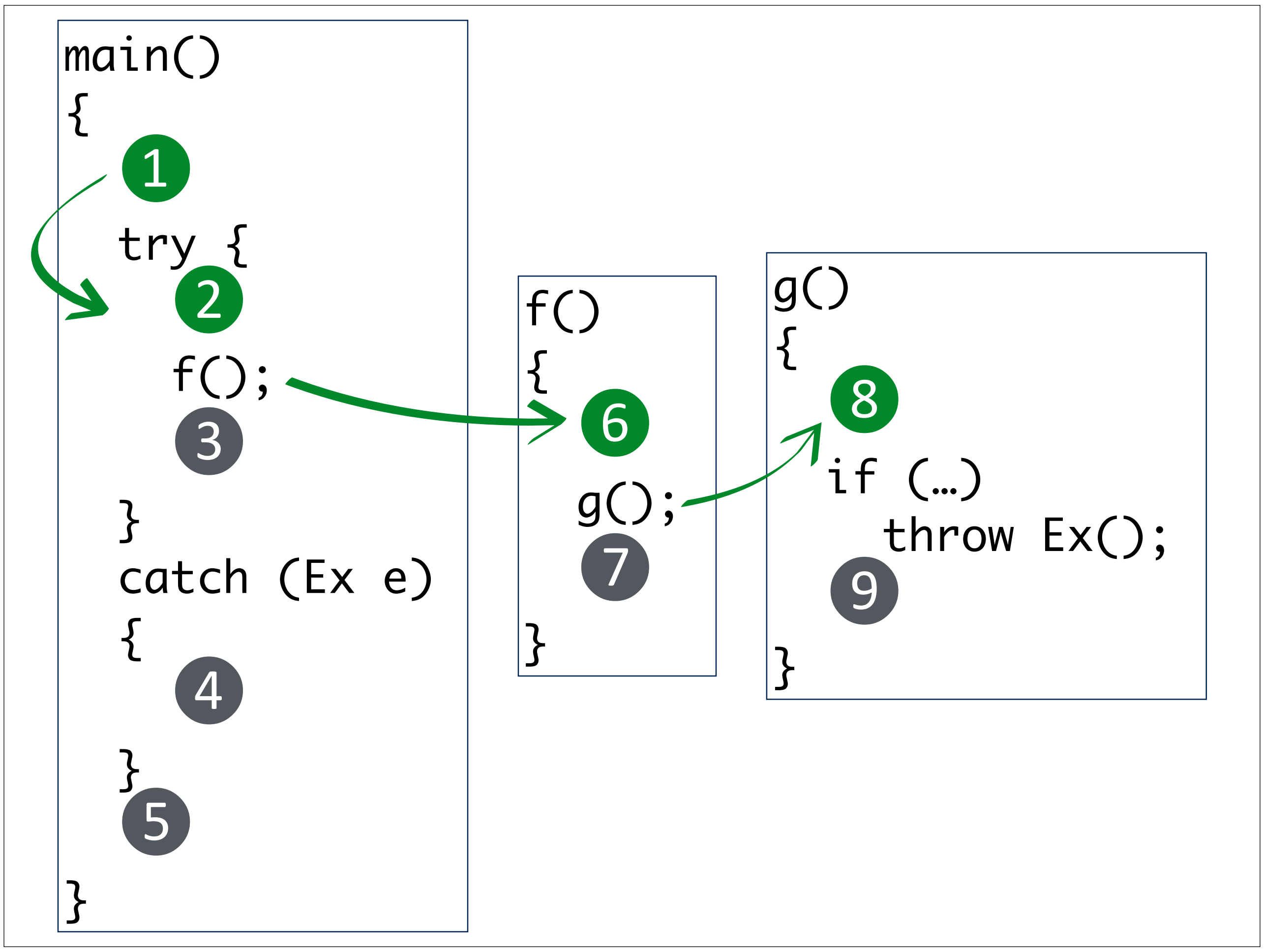
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

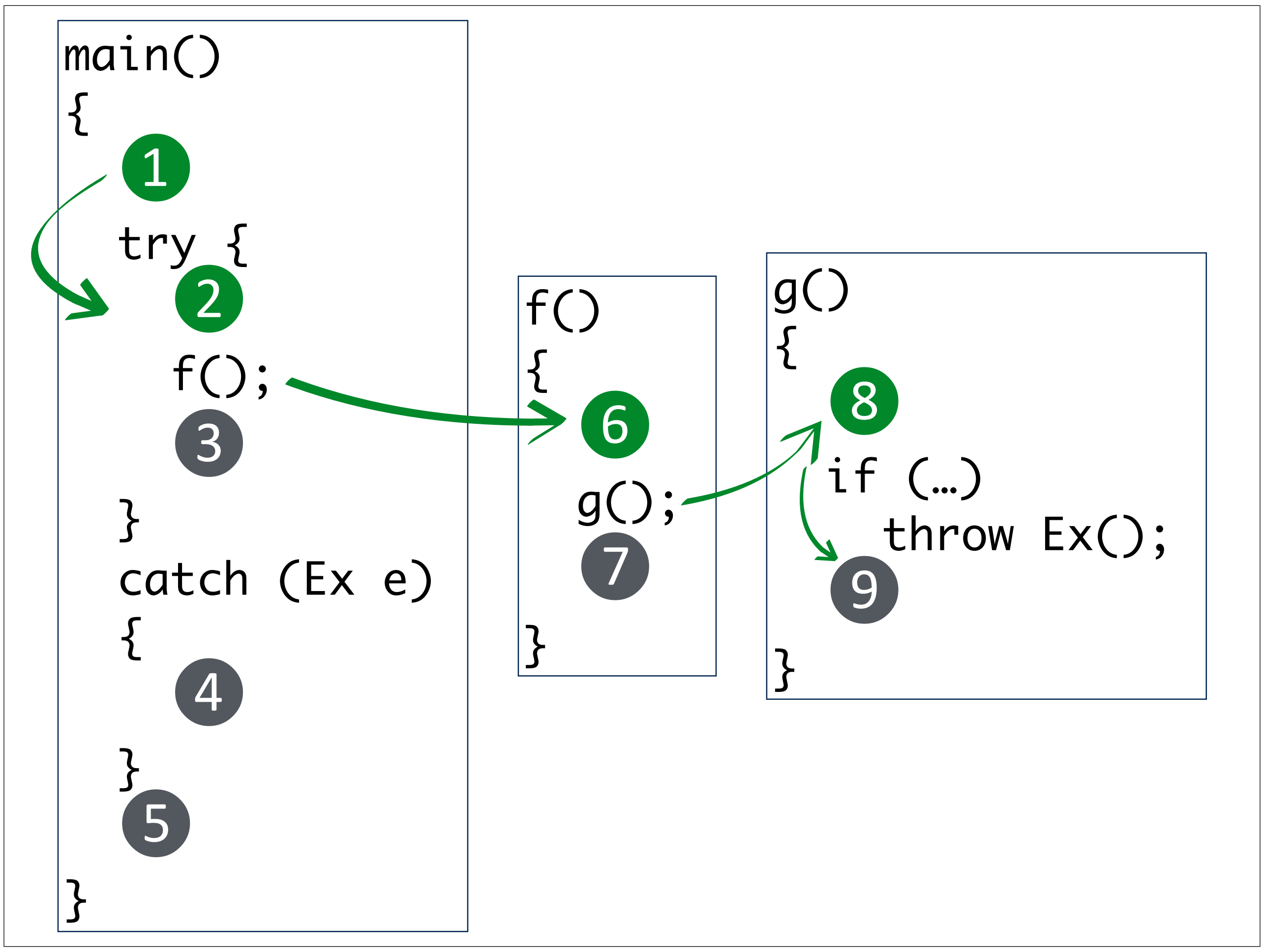
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

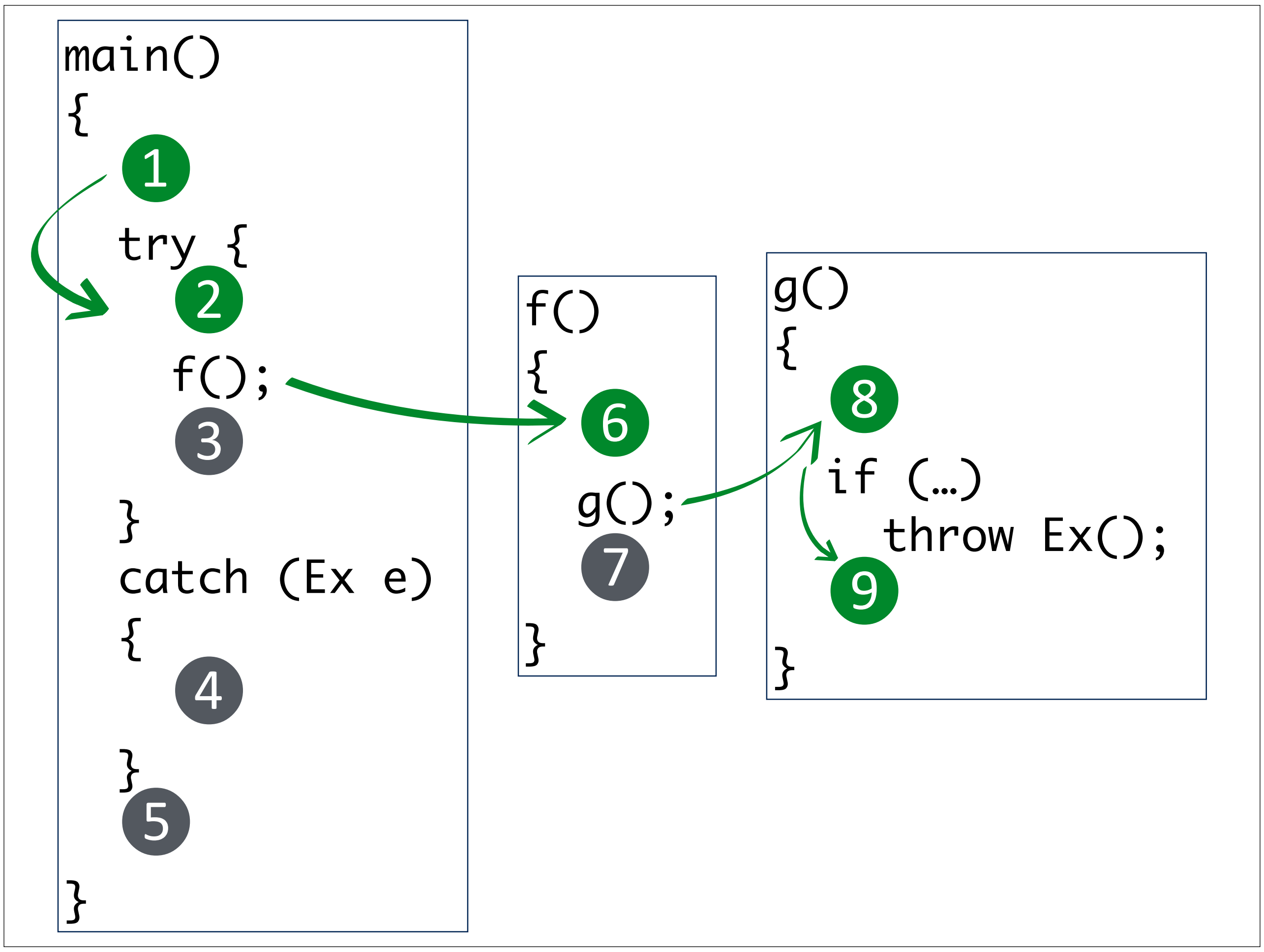
```
f()
{
  g();
}
```

```
g()
{
  if (…)
    throw Ex();
}
```


```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

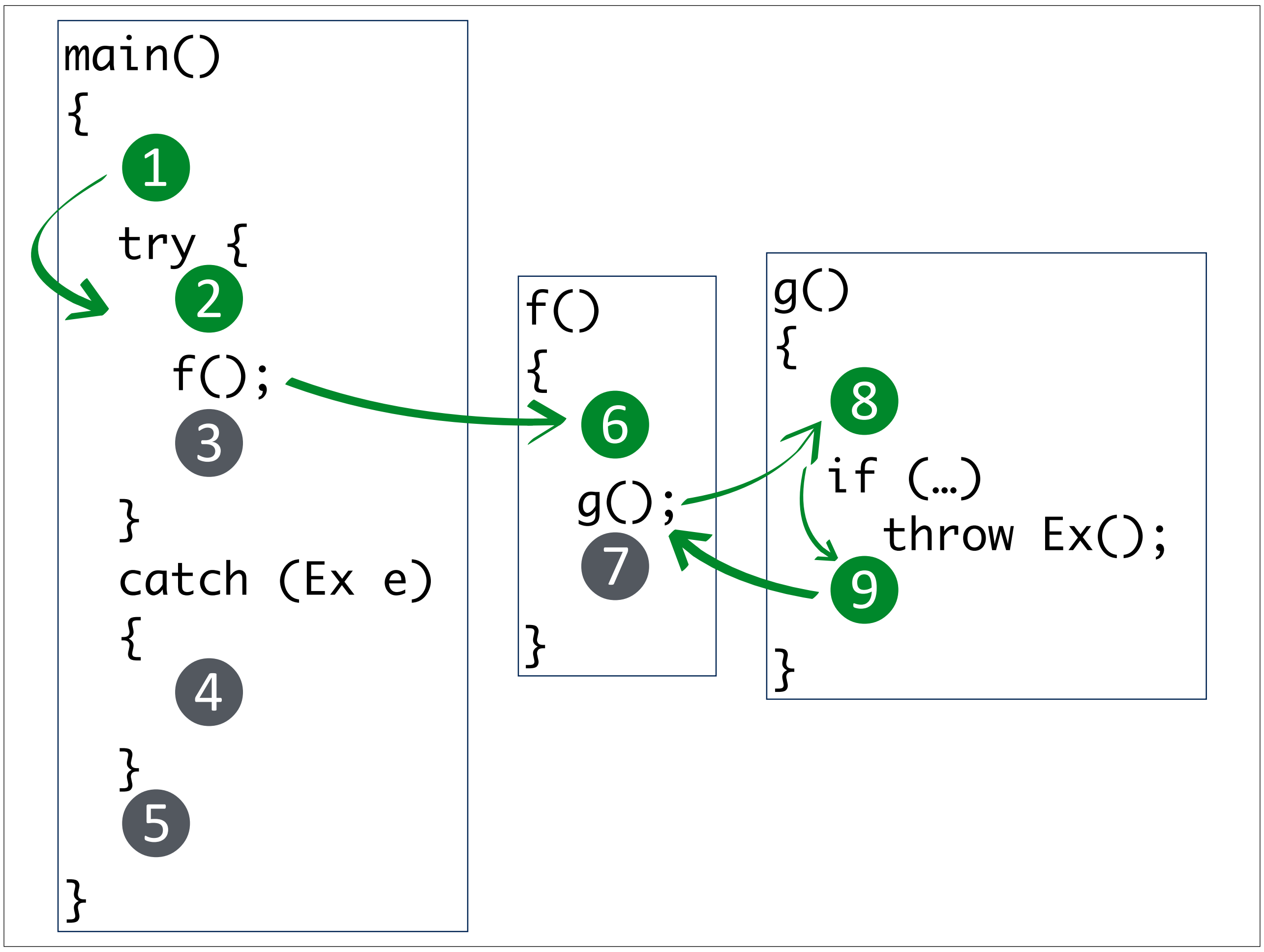
```
f()
{
  g();
}
```

```
g()
{
  if (…)
    throw Ex();
}
```


```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

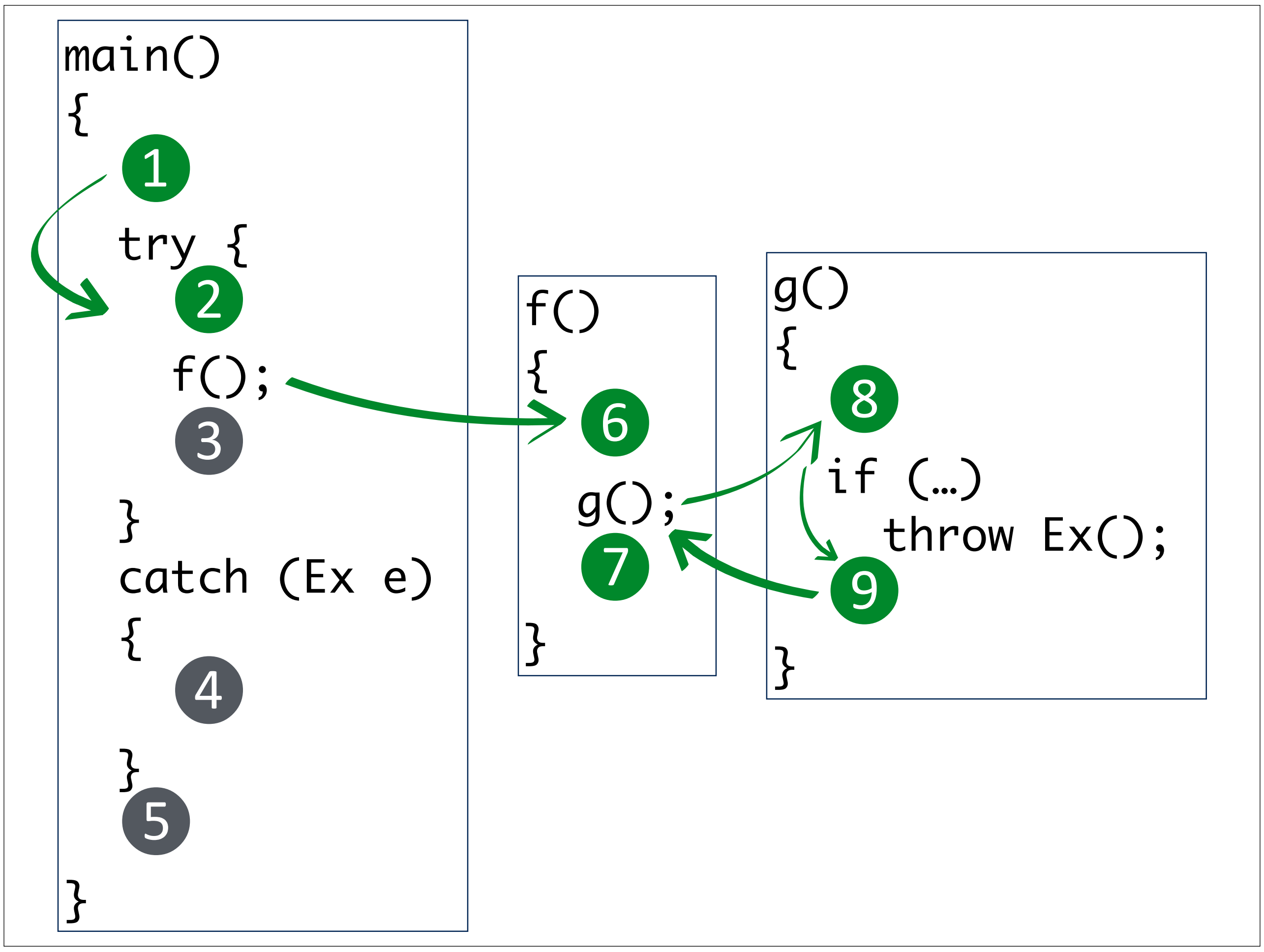
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

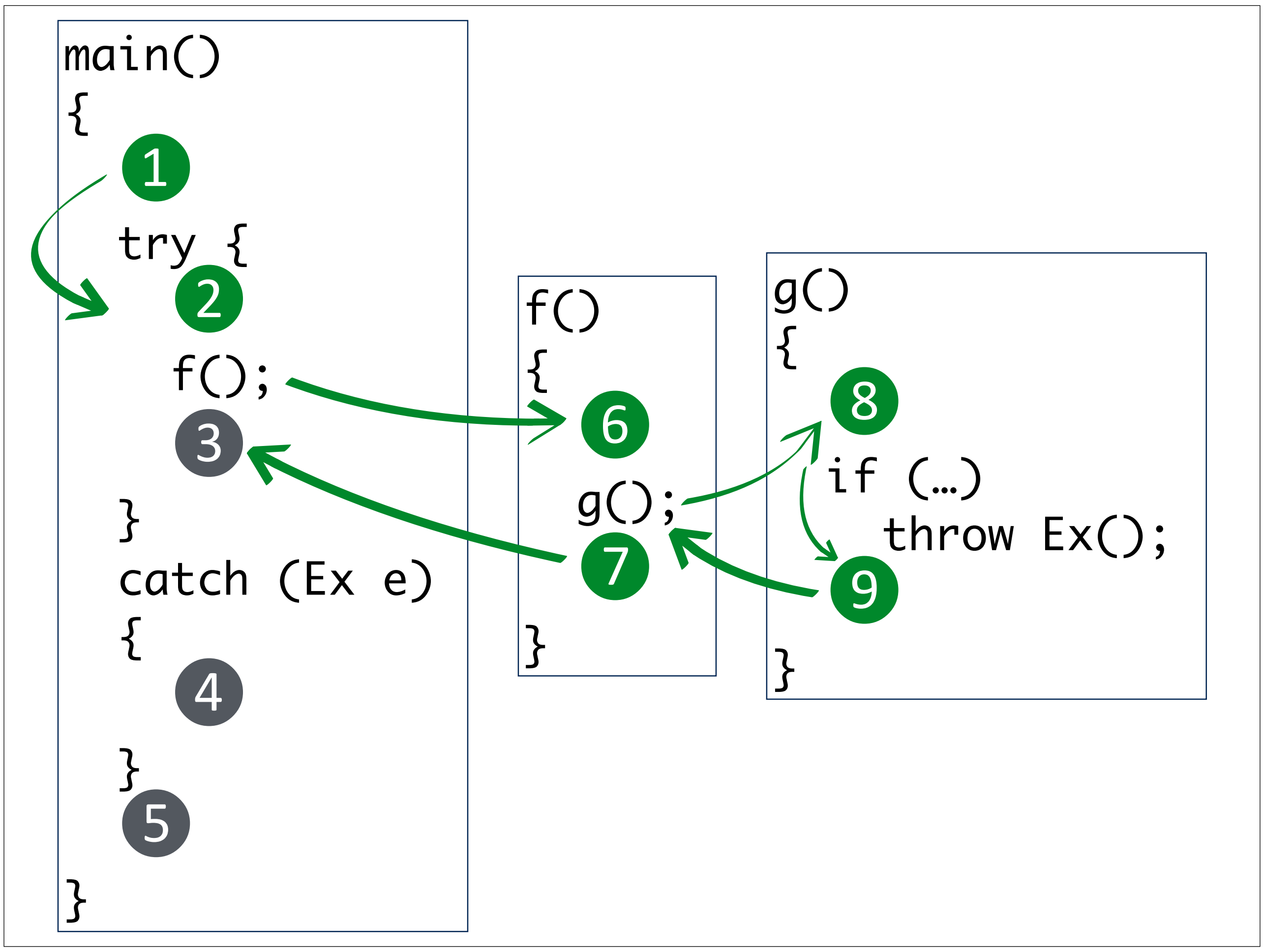
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

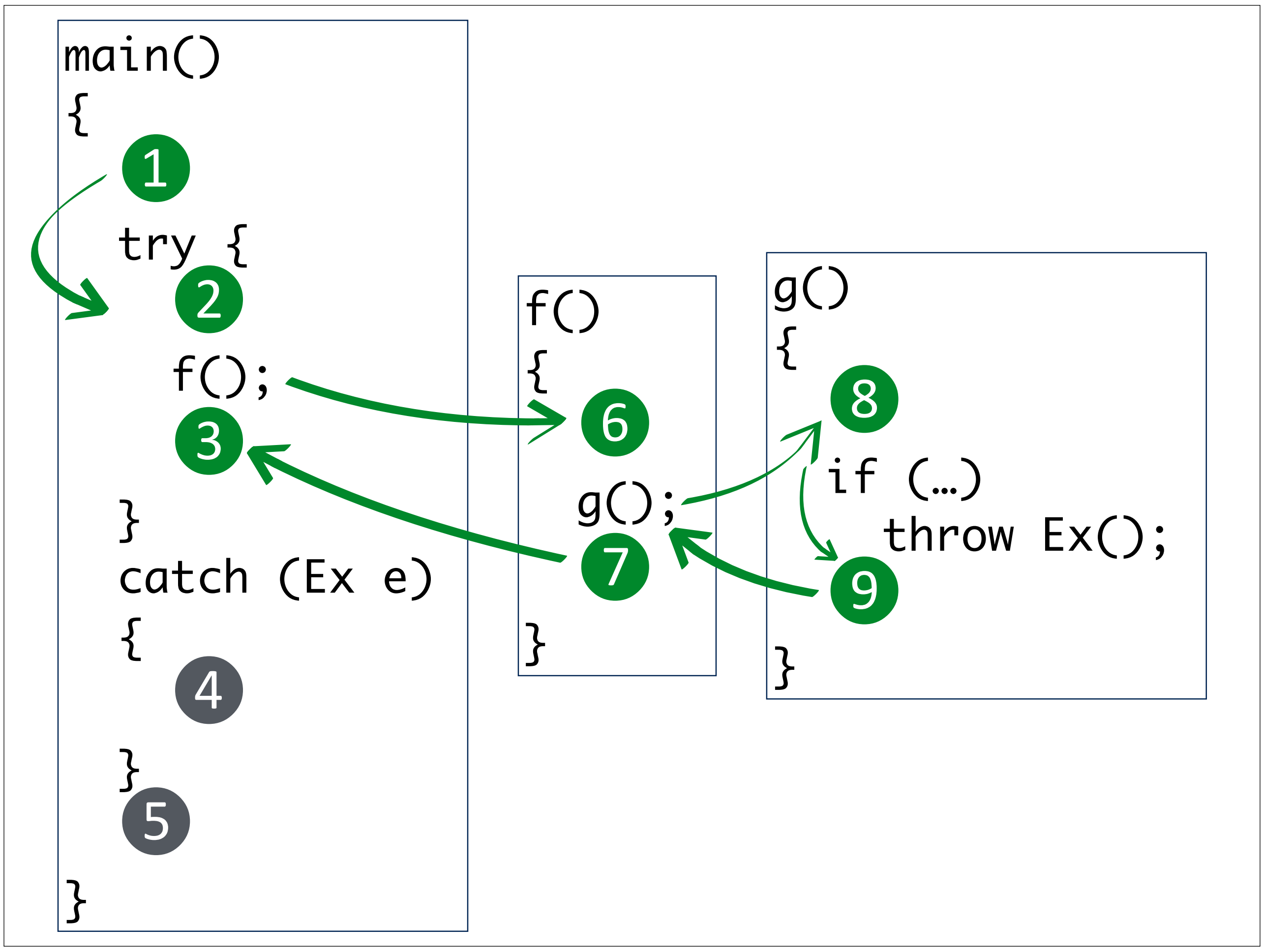
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

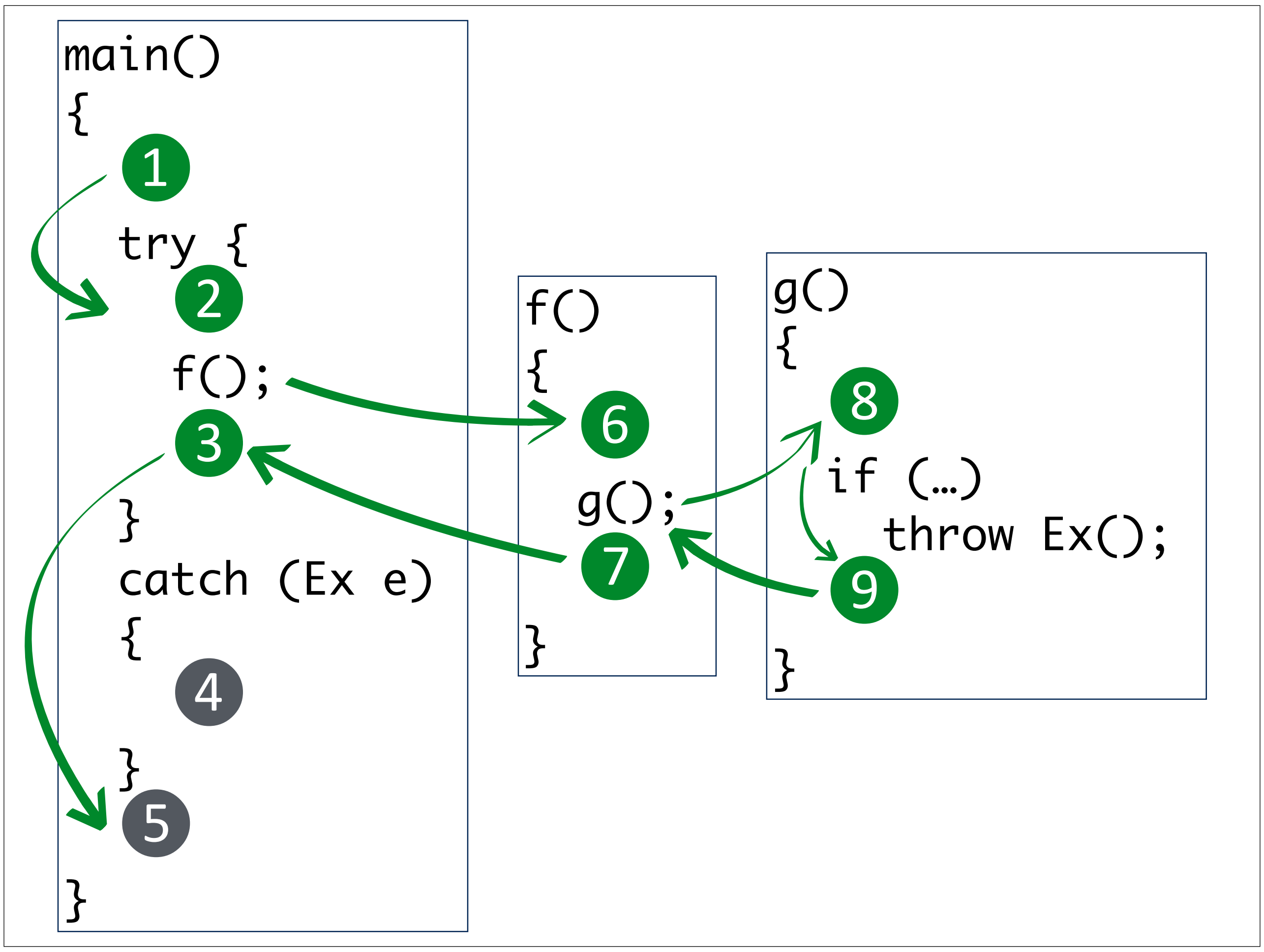
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

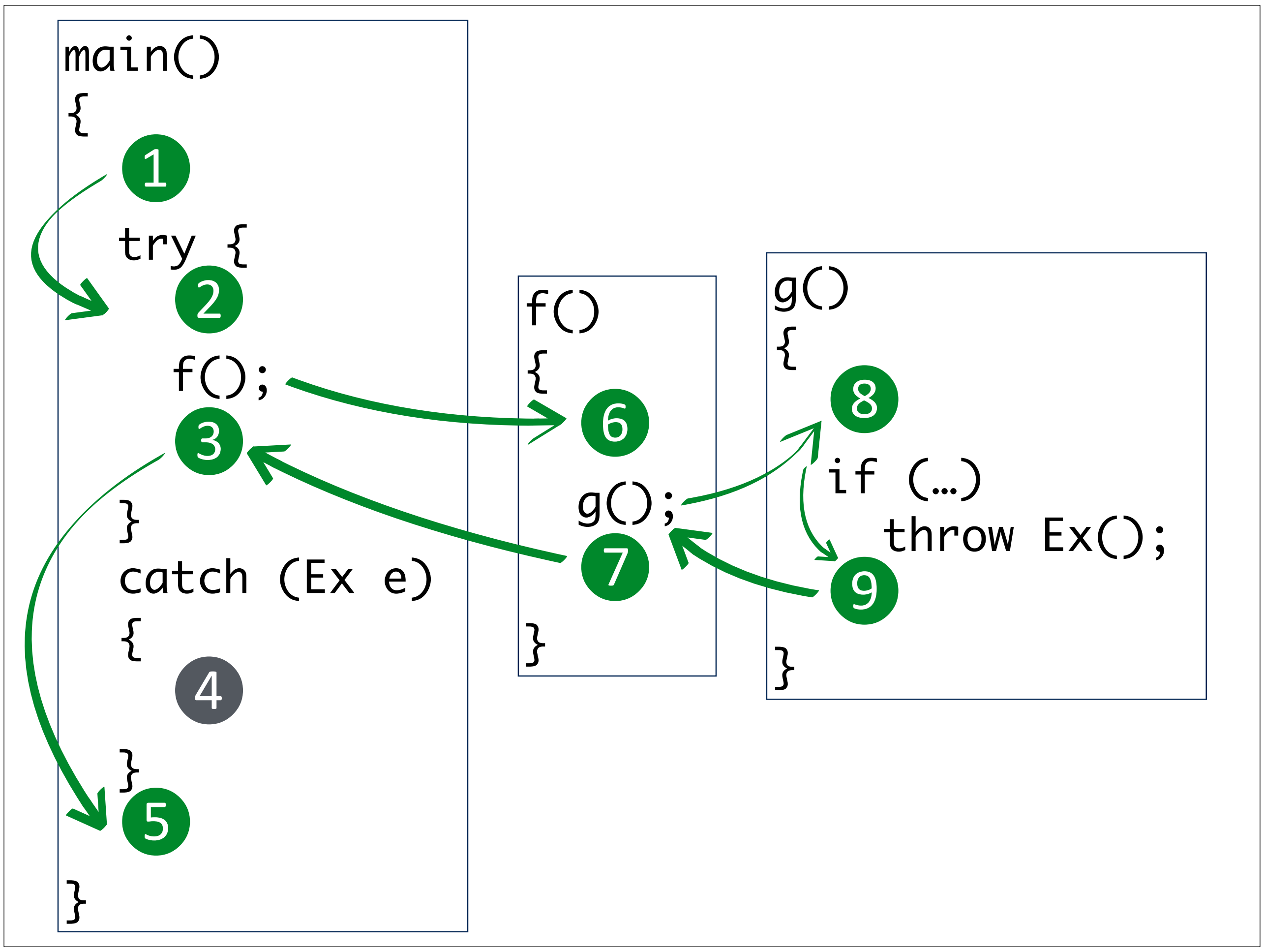
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

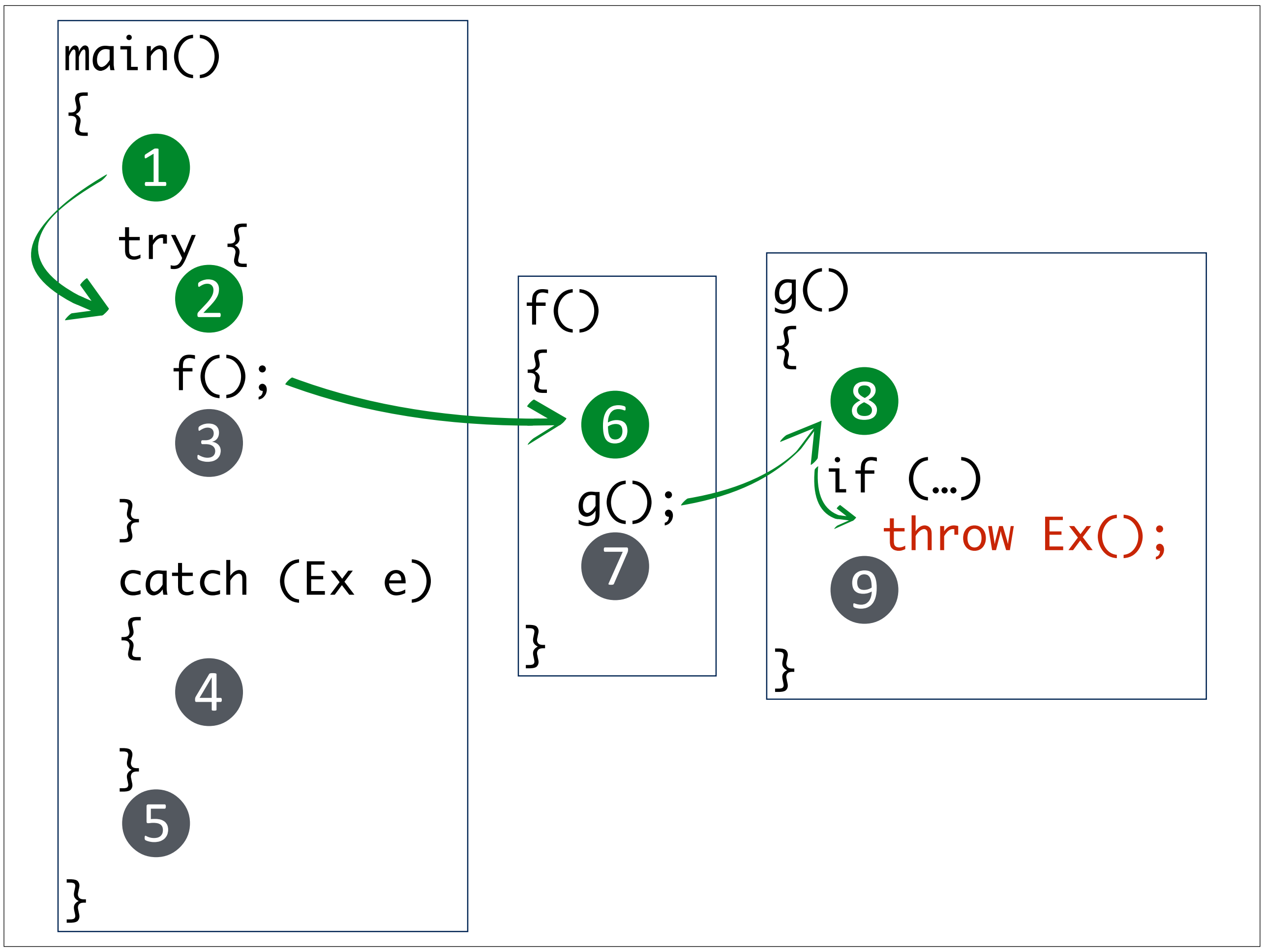
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

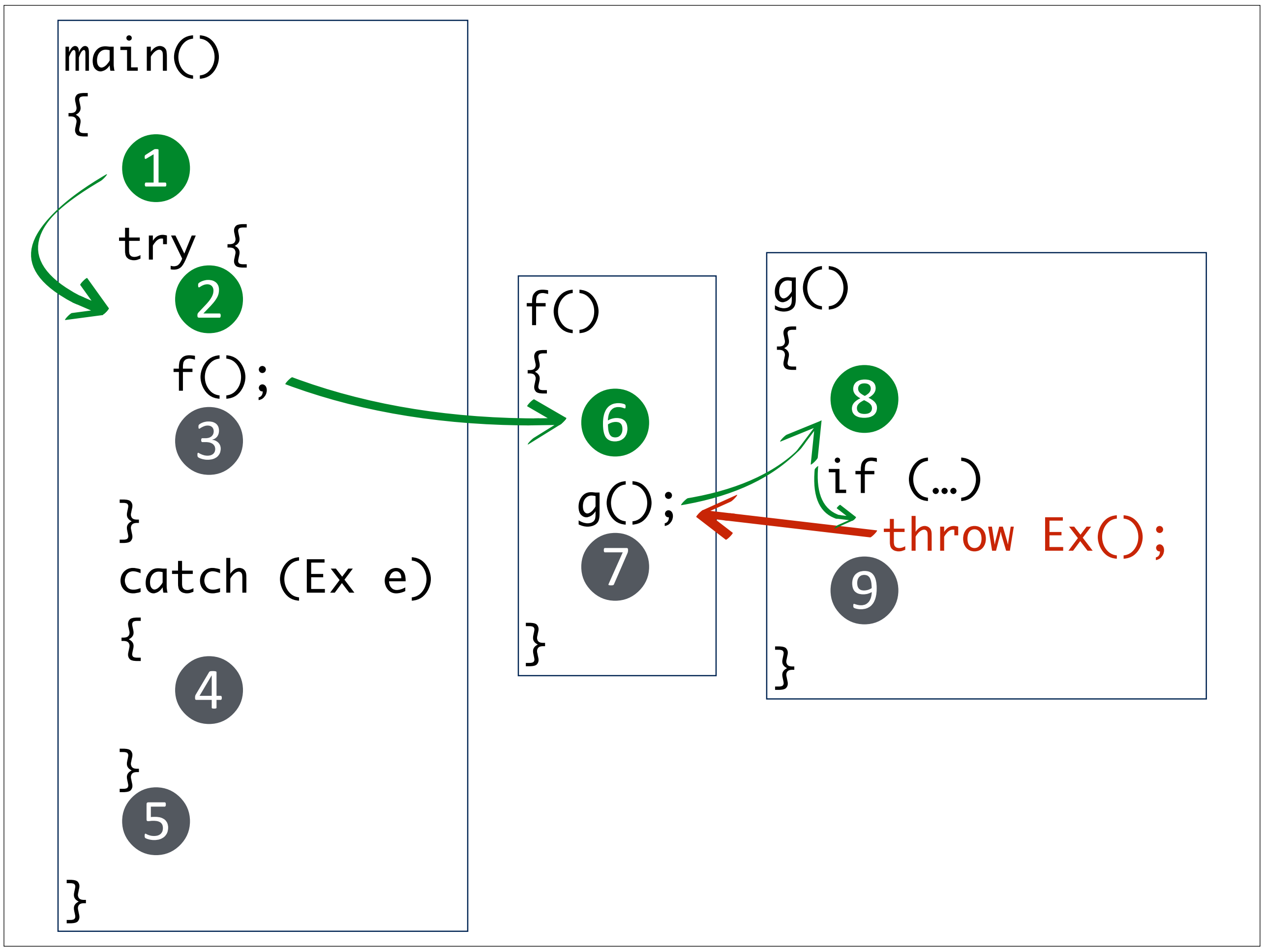
```
g()
{
  if (…)
    throw Ex();
}
```



```
main()
{
   try {
      f();
   }
   catch (Ex e)
   {
   }
}
```

```
f()
{
   g();
}
```

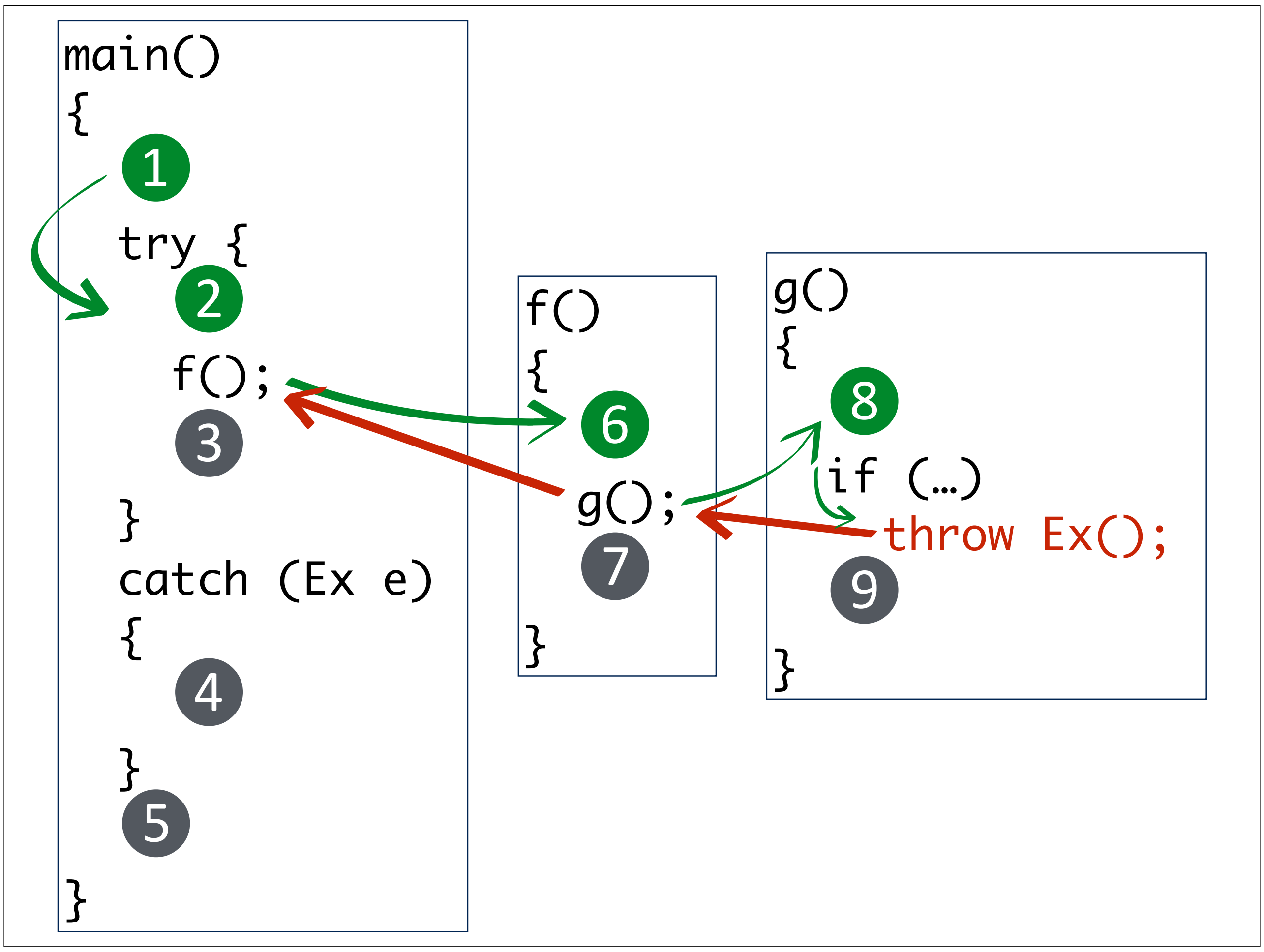
```
g()
{
   if (…)
      throw Ex();
}
```



```
main()
{
   try {
      f();
   }
   catch (Ex e)
   {
   }
}
```

```
f()
{
   g();
}
```

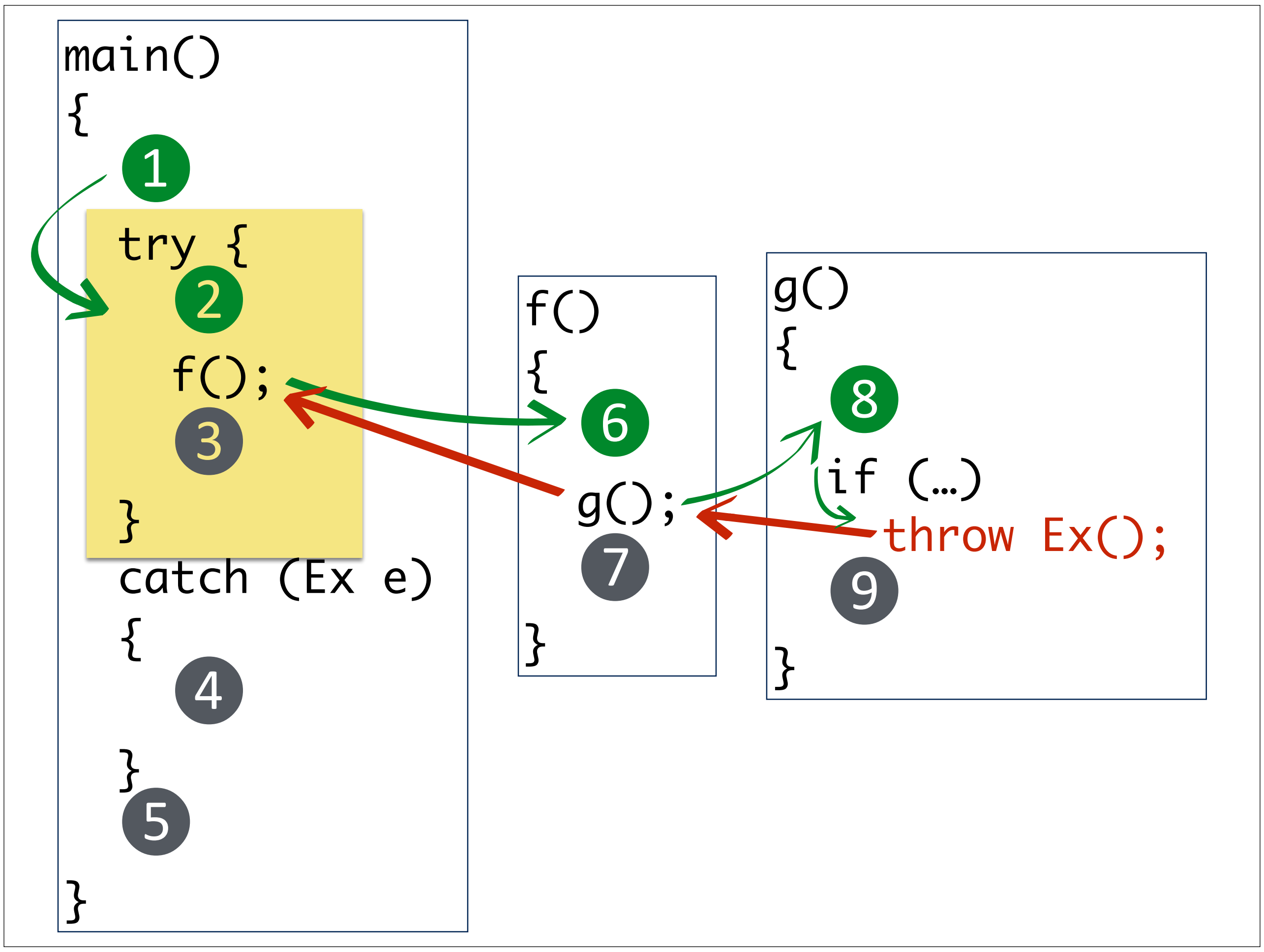
```
g()
{
   if (…)
      throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

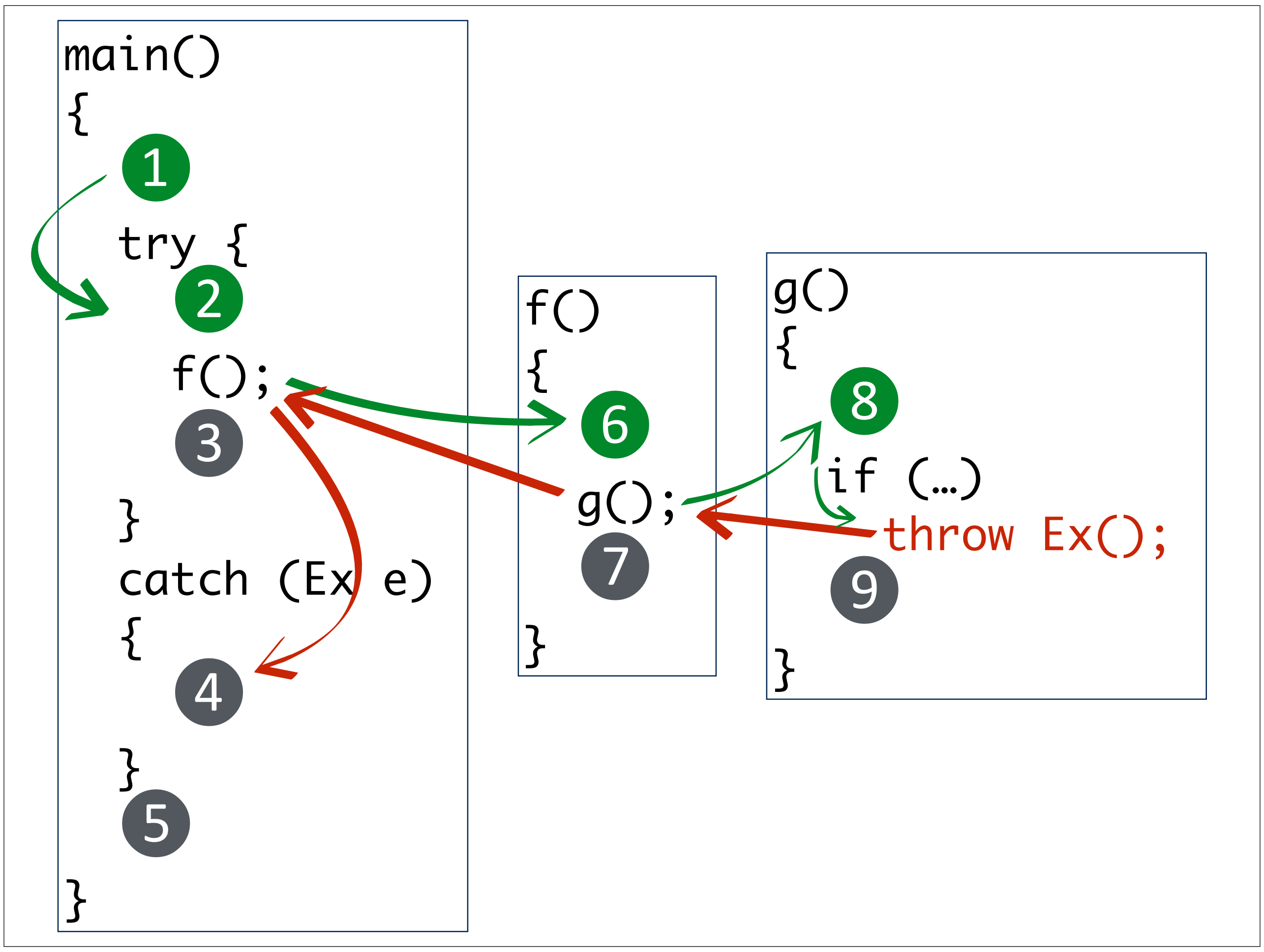
```
g()
{
  if (…)
  throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

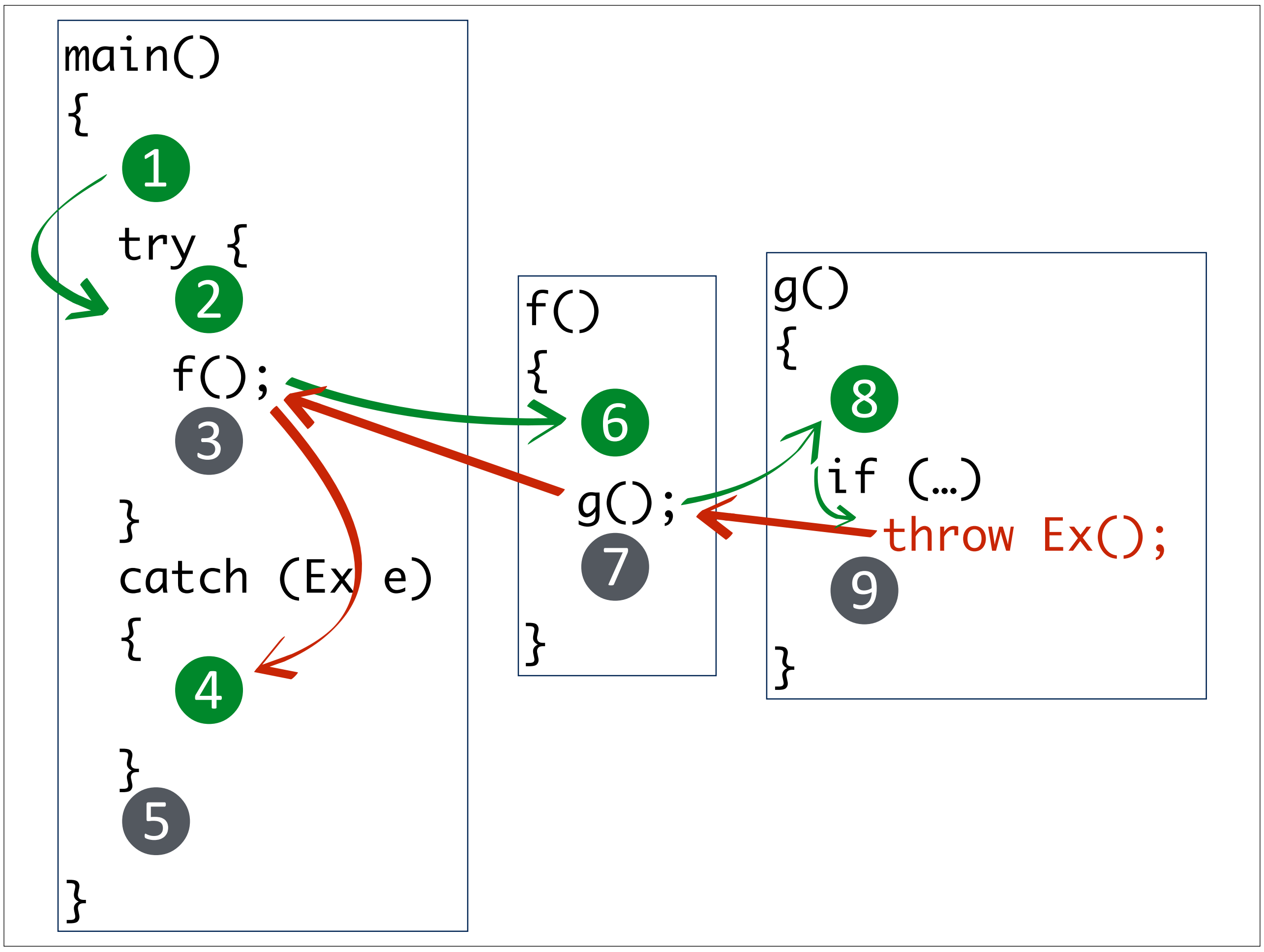
```
g()
{
  if (…)
  throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

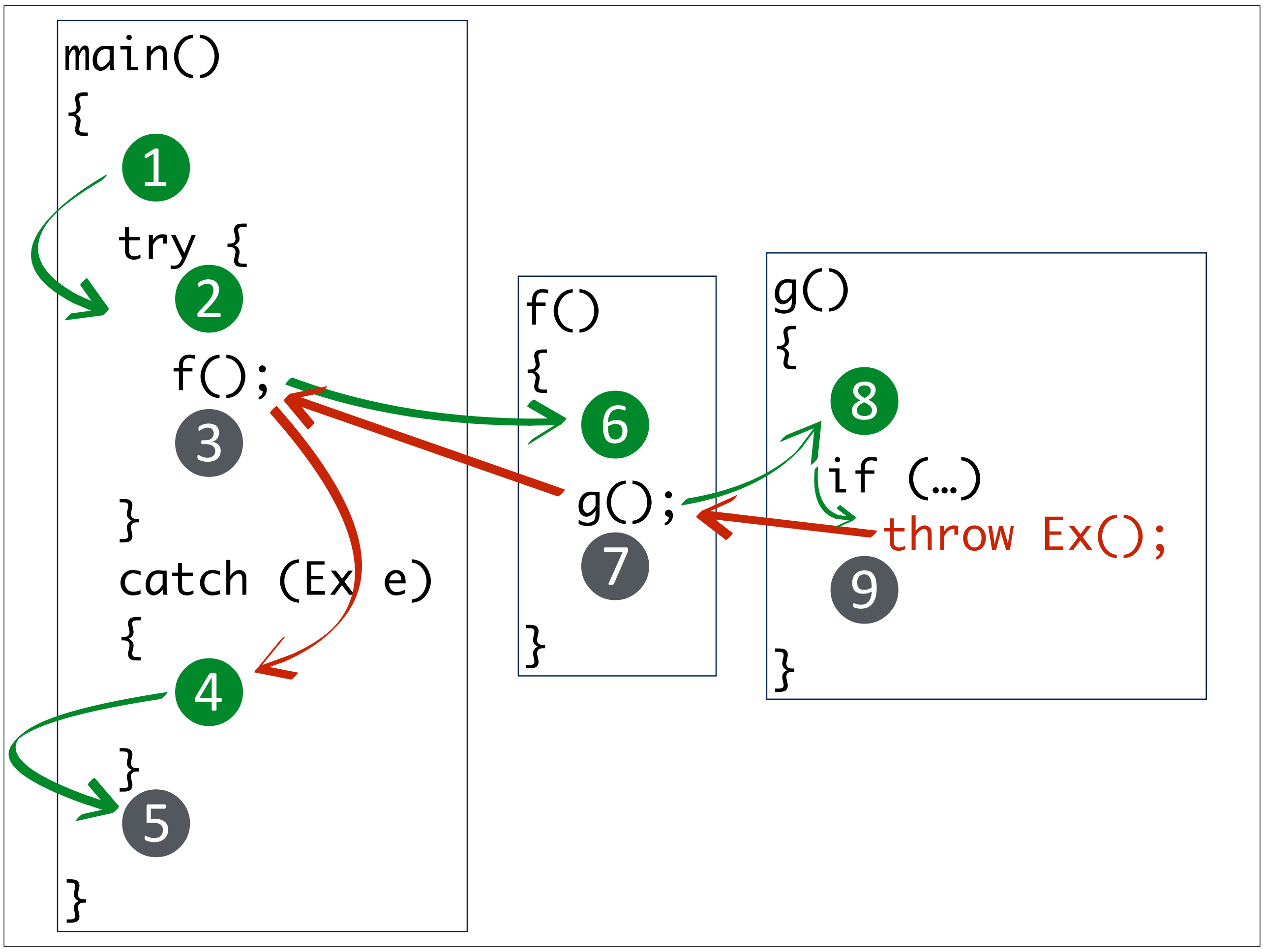
```
g()
{
  if (…)
  throw Ex();
}
```

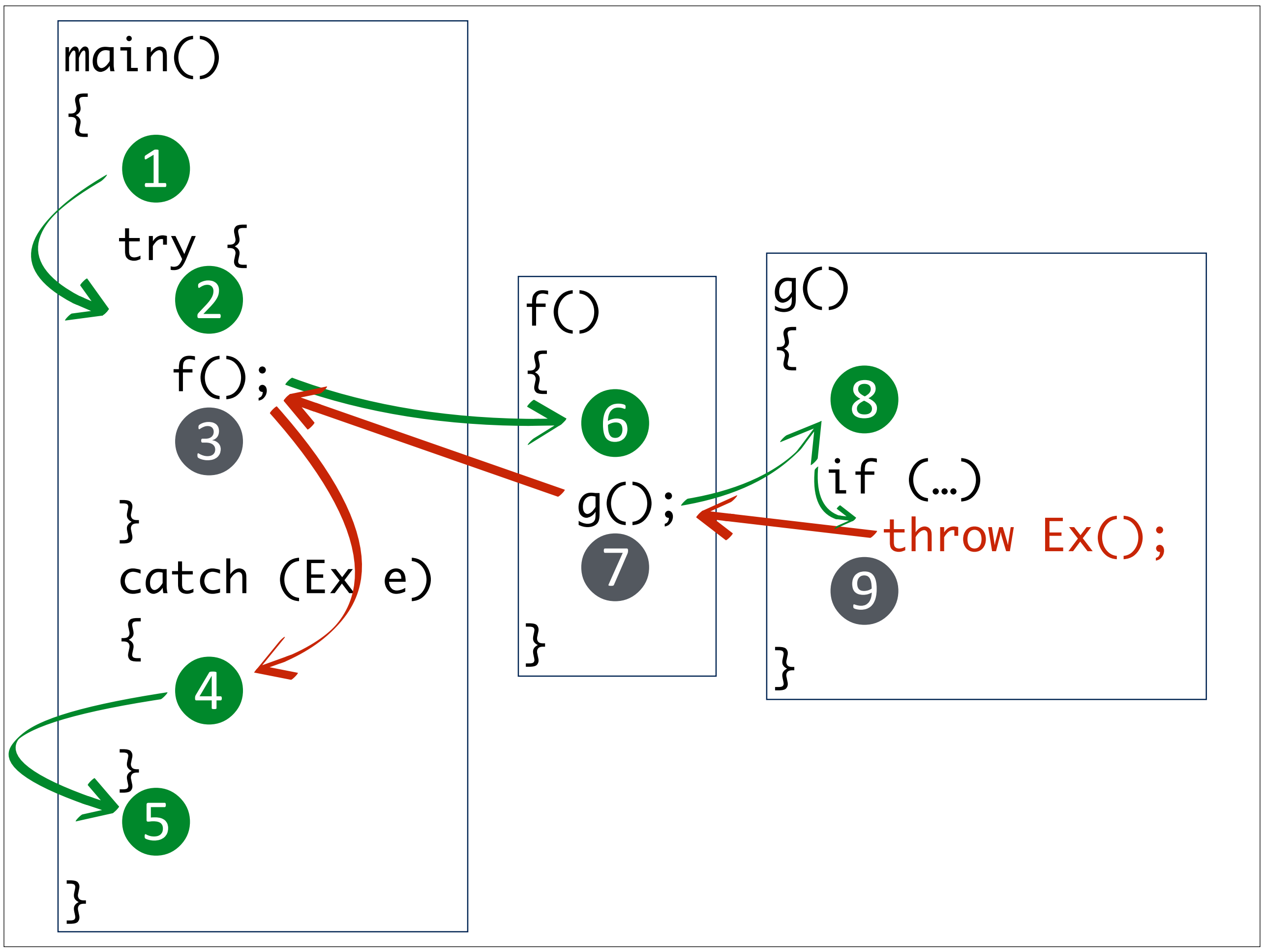


```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```

```
f()
{
  g();
}
```

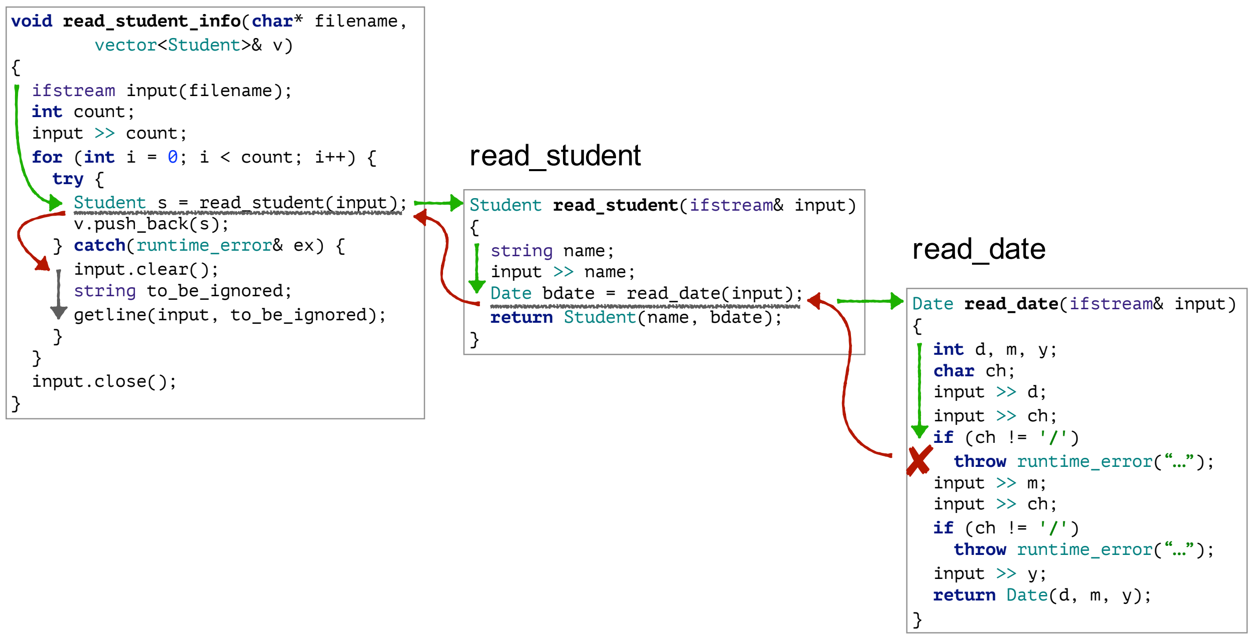
```
g()
{
  if (…)
  throw Ex();
}
```


```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```
```
f()
{
  g();
}
```
```
g()
{
  if (…)
  throw Ex();
}
```



```
main()
{
  try {
    f();
  }
  catch (Ex e)
  {
  }
}
```
```
f()
{
  g();
}
```
```
g()
{
  if (…)
  throw Ex();
}
```

read_student_info



```
void read_student_info(char* filename,
        vector<Student>& v)
{
  ifstream input(filename);
  int count;
  input >> count;
  for (int i = 0; i < count; i++) {
    try {
      Student s = read_student(input);
      v.push_back(s);
    } catch(runtime_error& ex) {
      input.clear();
      string to_be_ignored;
      getline(input, to_be_ignored);
    }
  }
  input.close();
}
```

read_student

```
Student read_student(ifstream& input)
{
  string name;
  input >> name;
  Date bdate = read_date(input);
  return Student(name, bdate);
}
```

read_date

```
Date read_date(ifstream& input)
{
  int d, m, y;
  char ch;
  input >> d;
  input >> ch;
  if (ch != '/')
    throw runtime_error("…");
  input >> m;
  input >> ch;
  if (ch != '/')
    throw runtime_error("…");
  input >> y;
  return Date(d, m, y);
}
```

```
void f(int i) {
    cout << "f1\n";
    if (i % 2)
        throw runtime_error("error");
    cout << "f2\n";
}

void g() {
    for (int i = 0; i < 5; i++) {
        cout << "g" << i << endl;
        f(i);
    }
}

void h() {
    try {
        cout << "h1\n";
        g();
        cout << "h2\n";
    } catch (runtime_error& ex) {
        cout << "h3\n";
    }
    cout << "h4\n";
}
```

بدون اجرای برنامه، تعیین کنید نتیجه فراخوانی h() چیست؟

```cpp
void f(int i) {
    cout << "f1\n";
    try {
        cout << "f2\n";
        if (i % 2)
            throw runtime_error("error");
        cout << "f3\n";
    } catch (runtime_error& ex) {
        cout << "f4\n";
        throw runtime_error("I insist!");
        cout << "f5\n";
    }
    cout << "f6\n";
}

void g() {
    try {
        cout << "g1\n";
        f(1);
        cout << "g2\n";
    } catch (runtime_error& ex) {
        cout << "g3\n";
    }
    cout << "g4\n";
}

void h() {
    try {
        cout << "h1\n";
        g();
        cout << "h2\n";
    } catch (runtime_error& ex) {
        cout << "h3\n";
    }
    cout << "h4\n";
}
```

بدون اجرای برنامه، تعیین کنید
نتیجه فراخوانی h() چیست؟