Declaring a Custom Exception Type

```
class MyException : public std::exception {
public:
    MyException(const std::string& msg) : message(msg) {}
    const char *what() const noexcept {
        return message.c_str();
    }
private:
    std::string message;
};
```

Nest Exception Declarations

Wrong:

```
class PersistentStorageFailure
        : public std::exception {
    ...
};

class PersistentStorage {
    ...
};



throw PersistentStorageFailure(
        “storage corrupted”);
```

Right:

```
class PersistentStorage {
public:
    class StorageFailure
            : public std::exception {
        ...
    };
    ...
};



throw PeristentStorage::StorageFailure(
        “storage corrupted”);
```

Create Useful Base Classes

```
class ExceptionWithErrno : public std::exception {
public:
    ExceptionWithErrno(const std::string& msg) {
        message = msg + ": " + strerror(errno);
    }
    const char *what() const noexcept {
        return message.c_str();
    }
private:
    std::string message;
};

class StorageError : public ExceptionWithErrno {
public:
    StorageError(const std::string& msg) : ExceptionWithErrno(msg) {}
}
```

Another Useful Base Class

```
class ExceptionWithPrintf : public std::exception {
public:
    ExceptionWithPrintf(const char *format, ...) {
         char buffer[1000];
         va_list args;
         va_start(args, format);
         vsnprintf(buffer, sizeof(buffer), format, args);
        message = buffer;
    }
    const char *what() const noexcept {
        return message.c_str();
    }
private:
    std::string message;
};
```

```
throw ExceptionWithPrintf("Couldn't bind to port %d: %s", portNum,
        strerror(errno));
```