Error Handling

Sergey Mechtaev
mechtaev@pku.edu.cn

Peking University

Error Handling

- What is an error?
- Two fundamental categories:
    - Bugs
        - Null dereference
        - Out-of-bounds array
    - Recoverable errors
        - Network connectivity problem
        - Parsing error
- Have to be treated differently!

Error Codes

Implementation

```
int foo() {
    // <try something here>
    if (failed) {
        return 1;
    }
    return 0;
}
```

Checking error code at call site

```
int err = foo();
if (err) {
    // Error!  Deal with it.
}
```

1. Inconvenient
2. May forget to check error code

Improved Error Codes (Functional Programming)

Implementation

```
fn bar() -> Result<(), Error>
  {
    let value = try!(foo);
    // Use value …
}
```

Checking error code at call site

```
fn bar() -> Result<(), Error> {
    match foo() {
        Ok(value) => /* Use
  value */,
        Err(err) => return Err(err)
    }
}
```

Forces user to check error

Checked Exceptions

Implementation

```
void foo() throws FooException,
                  BarException {
    …
}
```

Checking error code at call site

```
void bar() {
    try {
        foo();
    }
    catch (FooException e) {
        // Deal with FooException
    }
    catch (BarException e) {
        // Deal with BarException
    }
}
```

Issues with Exceptions

- Exceptions are used to communicate unrecoverable bugs, like null dereferences, divide-by-zero, etc.
- Java's RuntimeException are unchecked, thus not all exceptions is known in advance
- Complex/invisible control flow
- Exceptions make program slower

Anti-Pattern: Catching Generic Exception

```
try {
      doSomething();
} catch (Exception e) {
      // handle the exception
      log.error(e);
}
```

1. Handling is not bug-specific
2. Mixing bugs and recoverable errors

Anti-Pattern: Error Hiding

```java
public String readNameFromFile(Path file) throws IOException {
  String name = "";
  Charset charset = Charset.forName("US-ASCII");
  if (file != null) {
    try (BufferedReader reader =
           Files.newBufferedReader(file, charset)) {
      name = reader.readLine();
    } catch (Exception e) {
      System.err.println("error");
    }
  }
  return name;
}
```