Software Engineering: Tutorial 7

David Voigt

December 9th, 2022

Quiz 1: Question

```
...
else {
    try {
        throw new Exception
    } catch {
        case ex => println("something went wrong")
    }
}
```

Quiz 1: Answer

```
...
else {
    println("something went wrong")
}
```

Quiz 2: Question

```
def checkFormat(): Unit = {
    if (
        50 <= large.cost && large.cost <= 200 &&
        20 <= large.volume && large.volume <= 30 &&
        25 <= small.cost && small.cost <= 50 &&
        5 <= small.volume && small.volume <= 10
    )
        return
    else
        throw Exception("Problem is outside of constraints'
}
```

Quiz 2: Answer

```
def checkFormat(large: A, small: A): Bool = {
    50 <= large.cost && large.cost <= 200 &&
    20 <= large.volume && large.volume <= 30 &&
    25 <= small.cost && small.cost <= 50 &&
    5 <= small.volume && small.volume <= 10
}
```

Quiz 3: Question

```
def colourToHex(c: Colour): String = {
    var result = {
        c match {
            case Colour.Red => "FF0000"
            case Colour.Green => "00FF00"
            case Colour.Blue => "0000FF"
        }
    }
    return result
}
```

Quiz 3: Answer

```
def colourToHex(c: Colour): String =
    c match {
        case Colour.Red => "FF0000"
        case Colour.Green => "00FF00"
        case Colour.Blue => "0000FF"
    }
```

Quiz 4: Question

```
def applyToList[A, B](xs: List[A], f: A => B): List[B] = {
    var ys = List()
    xs.foreach(x => ys += f(x))
}
```

Quiz 4: Answer

```
xs.map(f)
```

Quiz 5: Question

```
def applyOptions[A, B](
    a: Option[A],
    b: Option[A],
    f: (A, A) => B
    ): Option[B] = {
    match a {
        case Some(x) => match b {
            case Some(y) => Some(f(x, y))
            case None => None
        }
        case None => None
    }
}
```

Quiz 5: Answer

```scala
def applyOptions[A, B](
    a: Option[A],
    b: Option[A],
    f: (A, A) => B
    ): Option[B] = {
    for
        x <- a
        y <- b
    yield f(x, y)
```

Interfaces

Example: Doors

- Most doors offer a common interface:
  - Open
  - Close
  - query state
- This interface is independent of how the door might be implemented:
  - Material
  - Hinges
  - Colour
- I do not need to know any of this in order to use it

Example: Doors in Scala

```
trait Door {
    def open(): Unit
    def close(): Unit
    def isOpen(): Bool
}
```

Example: Filesystems

- How does the filesystem on your computer work?
- . . . we do not really know, do we?
- Regardless, we can
  - drag and drop files to move them,
  - delete files,
  - and create new files
- All these operations work on your drive as well as with a USB stick, with potentially a different filesystem

Other Examples

- Public APIs: https://github.com/public-apis/public-apis
- Cat HTTP status codes API: https://http.cat