

# Basic of Java





Somkiat Puisungnoen

Search

Somkiat | Home

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro

Software Craftsmanship

Software Practitioner at สยามช่างนาฏกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ...

Java and Bigdata



somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc  
@somkiat.cc

Home Posts Videos Photos

Liked Following Share ...

+ Add a Button



# Agenda Day 1

- All about Java Programmer mistakes
- Object-Oriented Design
- SOLID
- Workshop



# Mistake of Java Programmer



# **Set JAVA\_HOME**



# How to build and compile ?



# Compile and Run

```
$javac -version  
$java -version
```



# How to read the Java document

## ?

<https://docs.oracle.com/javase/8/docs/>



# Build tools

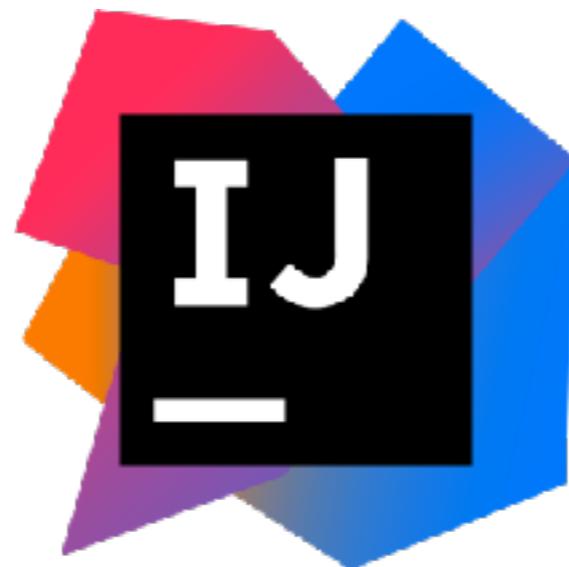


**Maven™**

 gradle

The Gradle logo consists of a green hexagonal icon with three curved segments in green, yellow, and red, followed by the word "gradle" in a large, lowercase, sans-serif font.

# IDE WARs



# Trust in your code ?



# Your code not break anything ?



# Write Unit tests

**JUnit**



# Hello with Unit testing



# **Basic of Java**



# Basic of Java

Working with String

if-else statement

Looping => for/for each, while and do-while



# Basic of Java

if-else statement

Looping => for/for each, while and do-while



# Workshop online



# CyberDojo

 **cyber-dojo**

**the place to practise programming**

**i'm on my own**

**we're in a group**

commercial use of [cyber-dojo.org](http://cyber-dojo.org) requires a licence

non-commercial use is free but

**please donate**

<http://www.cyber-dojo.org/>



# Day 2



# **Run your code/test in command line**



# **Check before run !!**

**For Windows OS**

```
$echo %JAVA_HOME%
```

**For Mac/Linux**

```
$echo $JAVA_HOME
```



# Run test with maven

\$mvnw clean test



# Build JAR file with maven

\$mvnw clean package



# Try to compile and win without Apache Maven



# Compile

```
$javac -s src/main/java -d your_target  
src/main/java/com/example/demo/HelloDay2.java
```

*-s = folder of source code*

*-d = folder of class files (destination)*



# Run

```
$java -cp your_target  
com.example.demo.HelloDay2
```

*-cp = classpath of class/JAR files*



# **Code Smell & Refactoring**

<https://sourcemaking.com/refactoring/smells>



# Code Smell



# Code Smell



*“Bad Code Smells are symptoms of poor design or implementation choices”*

[Martin Fowler]



# Code Smell Workshop

<https://github.com/emilybache/Tennis-Refactoring-Kata>



# Composition over Inheritance

[https://en.wikipedia.org/wiki/Composition\\_over\\_inheritance](https://en.wikipedia.org/wiki/Composition_over_inheritance)



# Composition over Inheritance

HAS-A

IS-A



# Source code management



# Download and Install Git

The screenshot shows the official Git website at <https://git-scm.com/>. At the top left is the Git logo with the tagline "distributed-is-the-new-centralized". A search bar is at the top right. The main content area features two columns of links: "About", "Documentation", "Downloads", and "Community". To the right, there's a diagram illustrating the distributed nature of Git with multiple repositories connected by red lines, and a large monitor displaying the latest source release information.

**git** --distributed-is-the-new-centralized

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

 Learn Git in your browser for free with [Try Git](#).

**About**  
The advantages of Git compared to other source control systems.

**Documentation**  
Command reference pages, Pro Git book content, videos and other material.

**Downloads**  
GUI clients and binary releases for all major platforms.

**Community**  
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release  
**2.18.0**  
Release Notes (2018-06-21)  
[Download 2.17.1 for Mac](#)

<https://git-scm.com/>



# Git command line

\$git status

\$git init

\$git add <file>

\$git commit -m “Reason to commit”

\$git push <remote> <local branch>

\$git pull <remote> <local branch>

\$git remote add <name> <url>



# Create repository at Github

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner      Repository name

 up1  

Great repository names are short and memorable. Need inspiration? How about [bookish-octo-tribble](#).

---

Description (optional)

---

 **Public**  
Anyone can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

---

**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

---

Add .gitignore: **None** ▾ | Add a license: **None** ▾ 

---

**Create repository**



# Create already

The screenshot shows a GitHub repository page for 'up1 / sample'. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. A 'Unwatch' button with a count of 1 is also present. The main content area features a 'Quick setup — if you've done this kind of thing before' section with options to 'Set up in Desktop' or choose between HTTPS and SSH, providing the URL <https://github.com/up1/sample.git>. Below this, a note says 'We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#)'. Further down, instructions for creating a new repository on the command line are shown in a red-bordered box:

```
echo "# sample" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/up1/sample.git
git push -u origin master
```



# Working with Git



# 1. Create repos on your folder

```
$git init
```



## 2. Add file/folder to git repos

\$git add <filename>

\$git add <folder>



# 3. Commit your change

```
$git commit -m "<message>"
```



# 4. Add remote repository

\$git remote add <name> <remote url>

\$git remote add **origin** **https://github.com/up1/sample.git**



# 5. Push your change to remote repos

```
$git push origin master
```

And check your code at [github.com](https://github.com)



# Day 3



# SOLID principle



# SOLID principle



**SOLID**

Software development is not a Jenga game.



# 1. Single Responsibility Principle

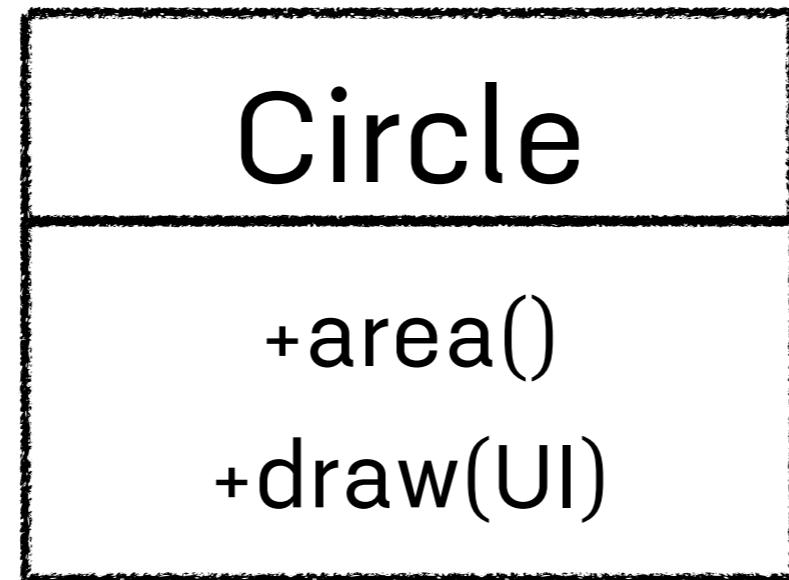


## Single Responsibility Principle

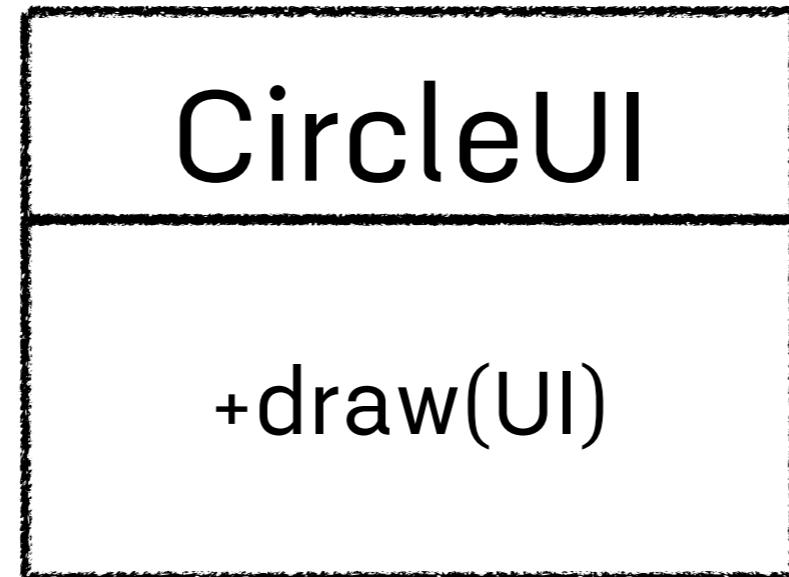
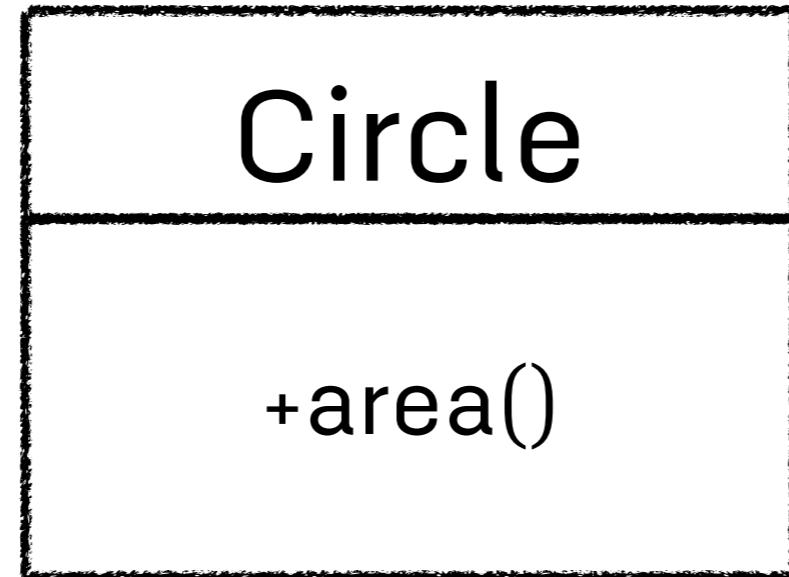
Just because you *can* doesn't mean you *should*.



# Example



# Example



## 2. Open-Closed Principle



# Example

## DrawingEditor

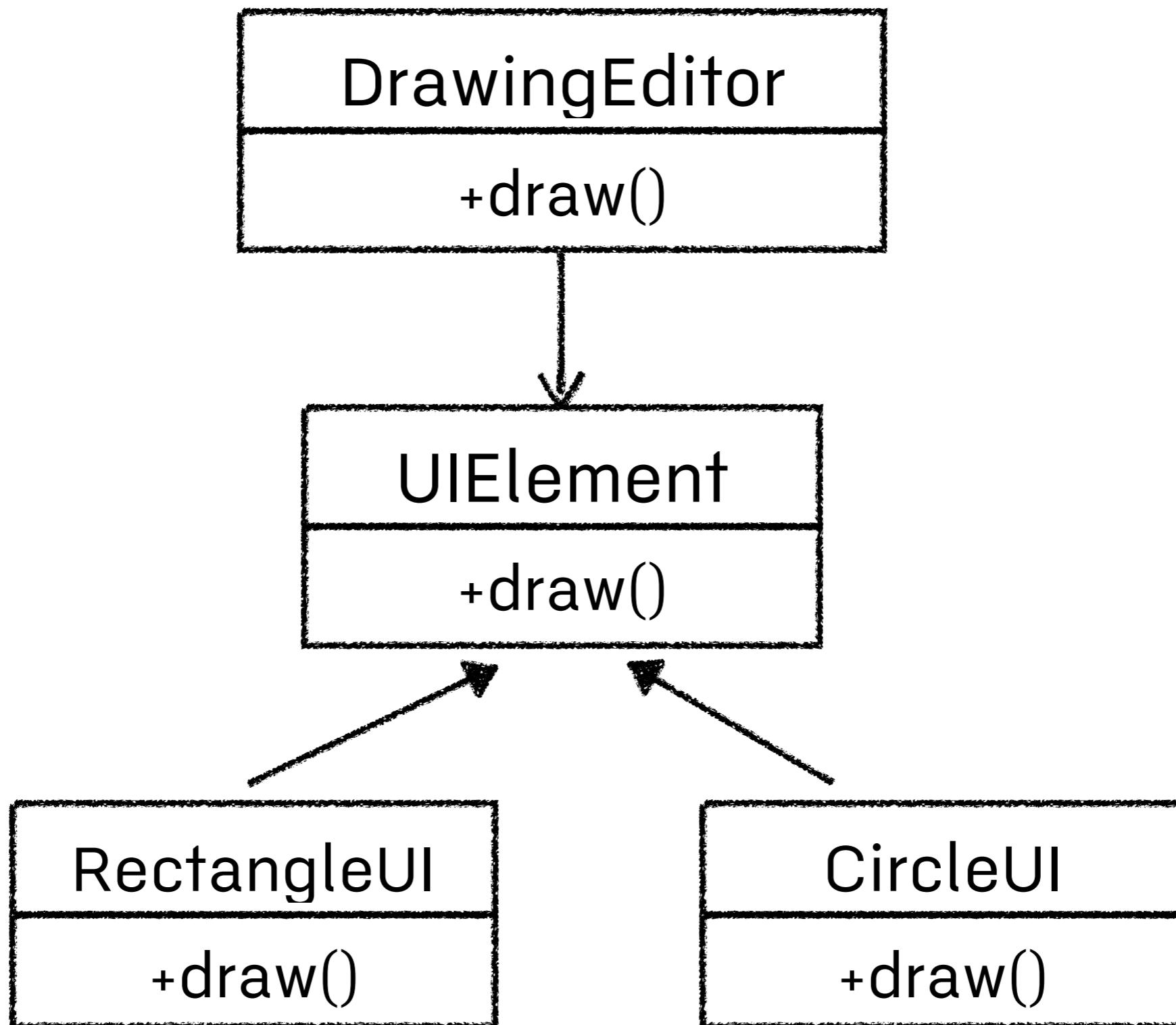
+draw()

-drawCircle()

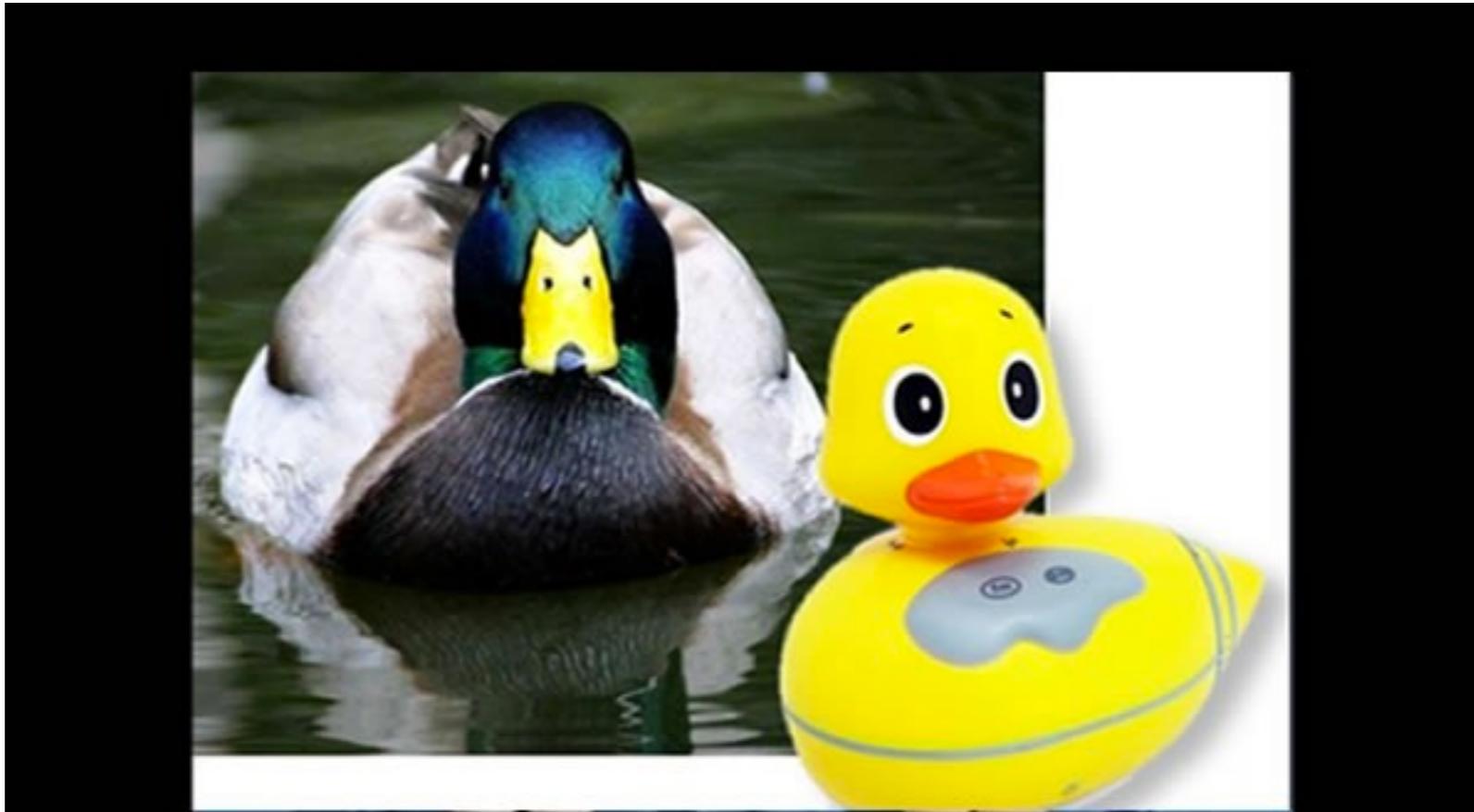
-drawRectangle()



# Example



# 3. Liskov Substitution Principle

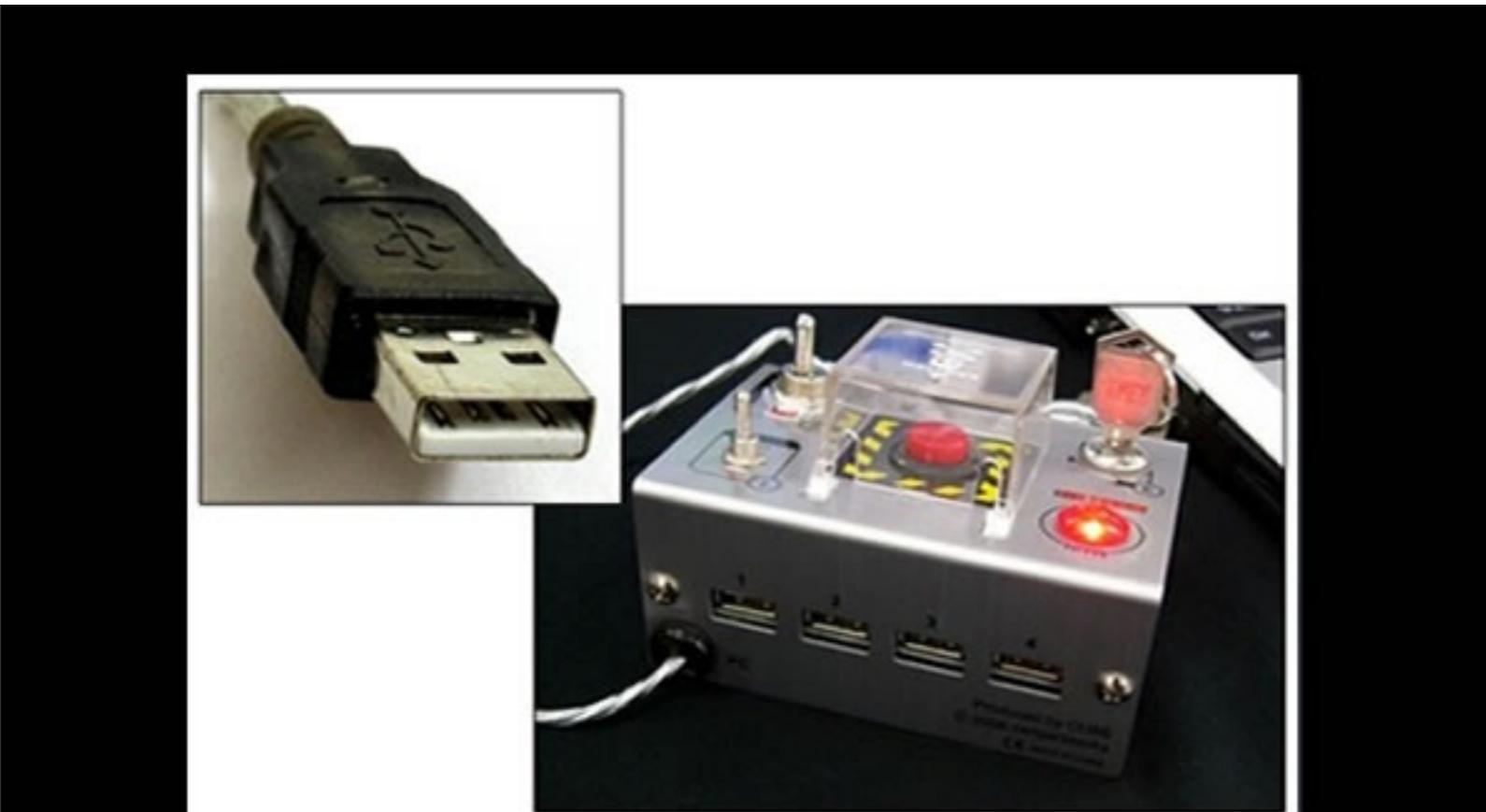


## Liskov Substitution Principle

If it looks like a duck and quacks like a duck but needs batteries,  
you probably have the wrong abstraction.



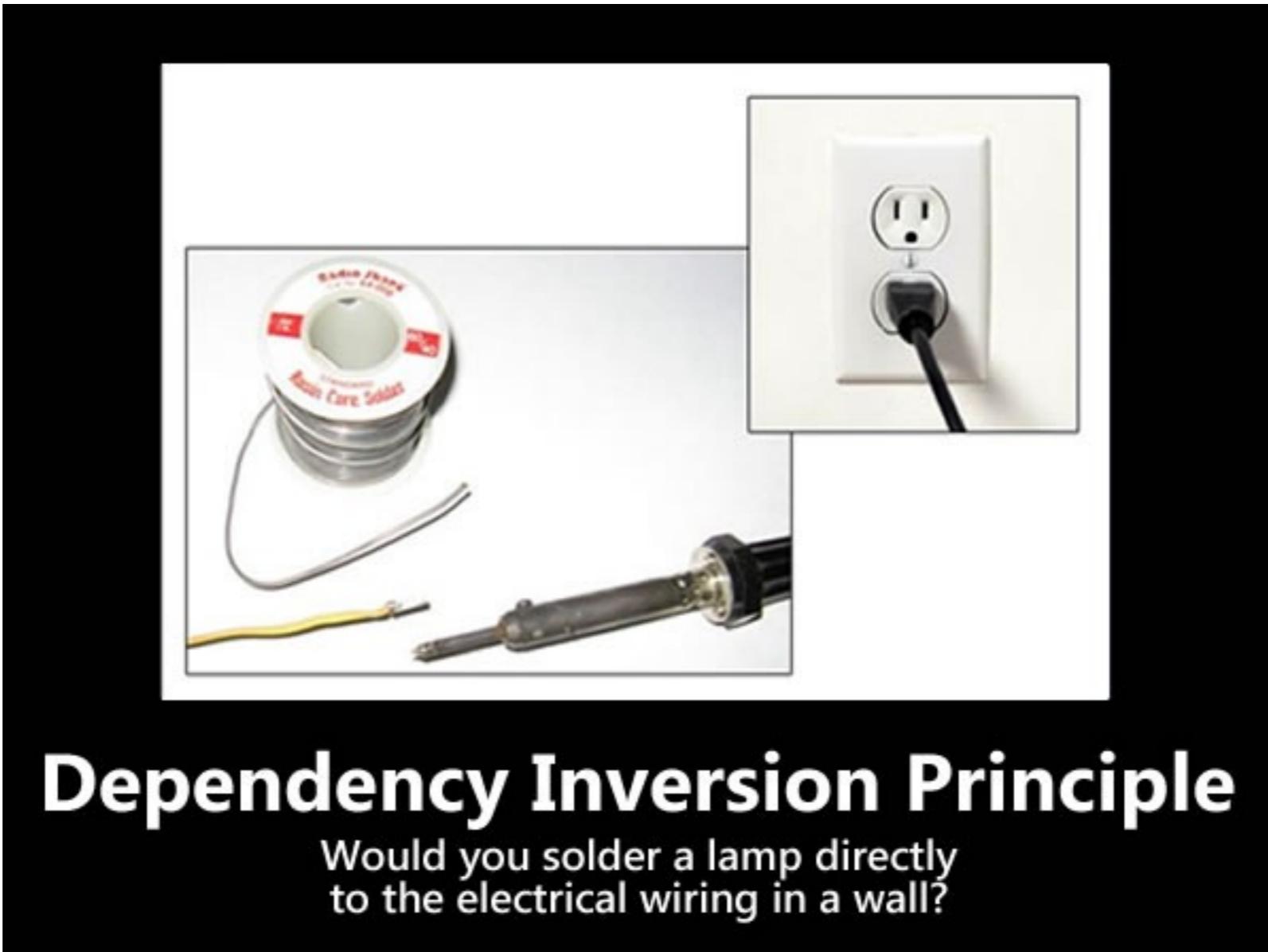
# 4. Interface Segregation Principle



**Interface Segregation Principle**  
You want me to plug this in *where?*



# 5. Dependency Inversion Principle

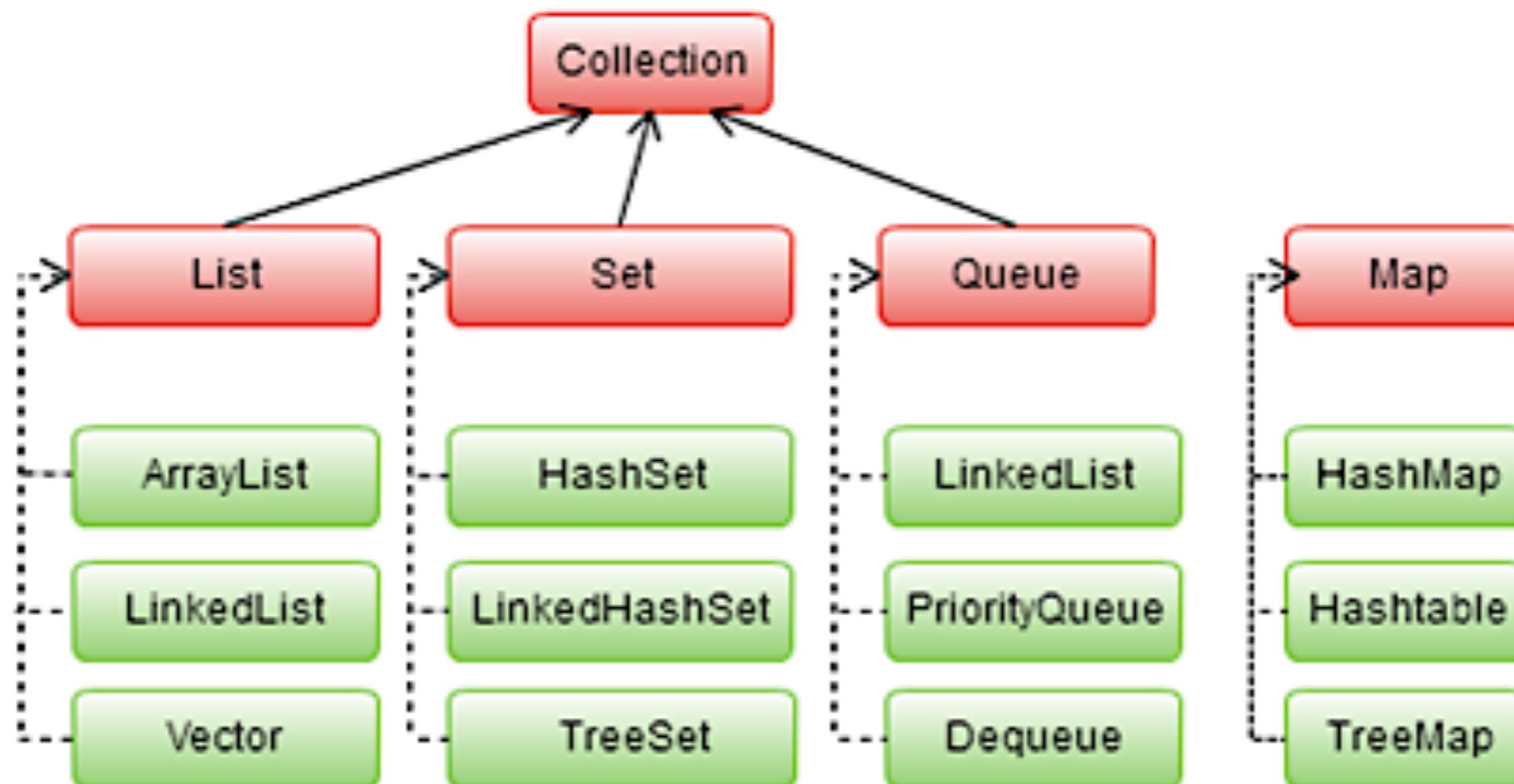


# Java Collection Framework

<https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html>



# Java Collection Framework



# Java 8 working with Stream



# Types of Error in Java

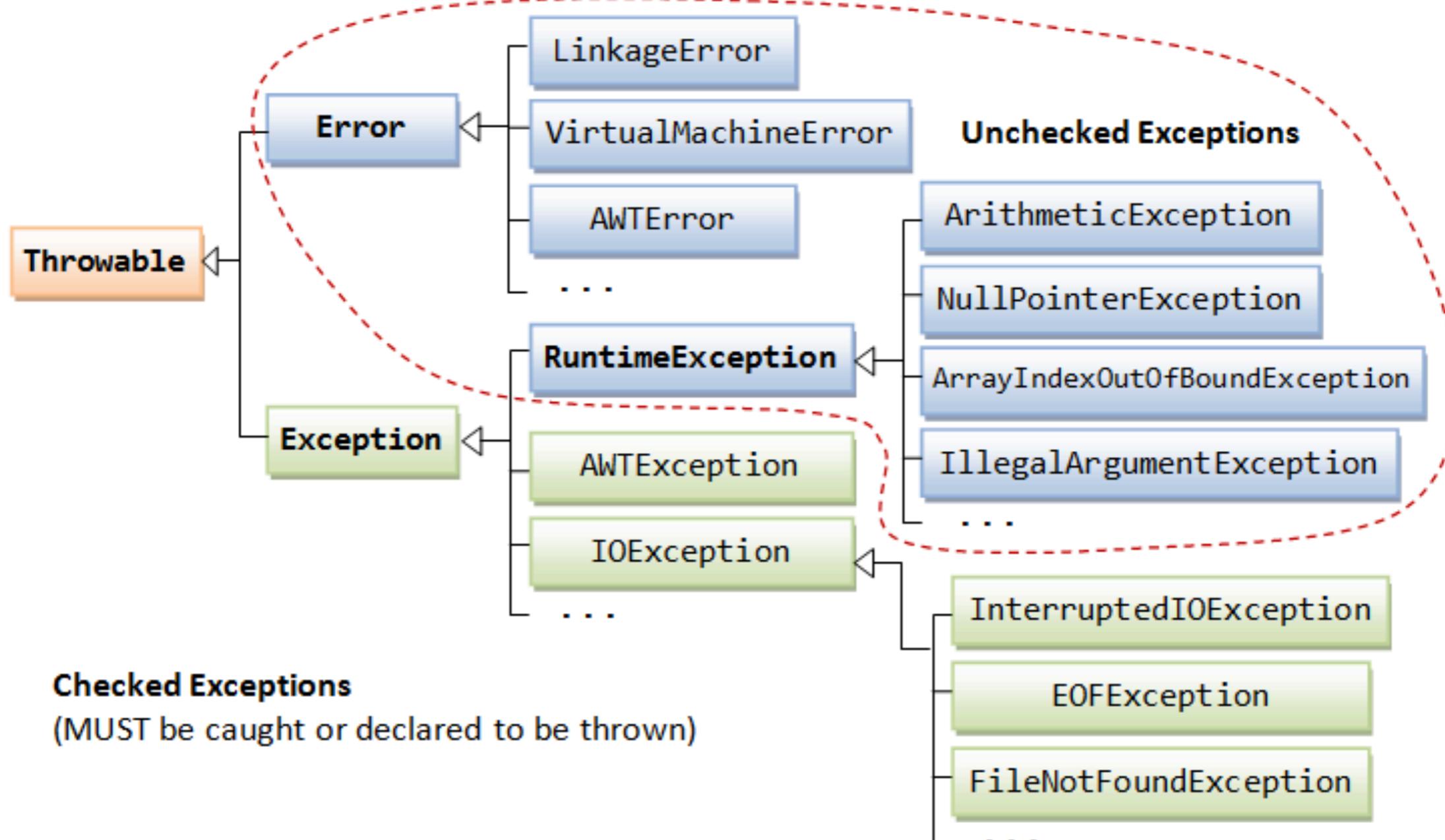


# Types of Errors

- Compile-time errors
- Runtime errors
- Logical errors



# Exception classes



# Java Developer should learn 2018

<https://dzone.com/articles/5-things-java-programmer-should-learn-in-2018>



# Coding everyday (2 hours)



# Participate coding challenges



# Try new version of Java



# Learn Java performance tuning



# Learn Spring Framework



# **Write Unit tests**



# Profiling your app once a month



# Learn new JVM languages

