Visibility, Static, and Exception Handling

June 19, 2017

Reading Quiz

What is Printed?

```
System.out.print("A");

try {

   System.out.print("B");
   // SomeExceptionType is thrown

} except (SomeExceptionType e1) {

   System.out.print("C");

} except (OtherExceptionType e2) {

   System.out.print("D");

} finally {

   System.out.print("E");

}

System.out.print("F");
```

A. "ABCDEF"

B. "ABF"

C. "ABCF"

D. "ABEF"

E. "ABCEF"

What is the Purpose of Exceptions?

A. To indicate a performance issue

B. To indicate an uncommon error has occurred, that the programmer should handle

C. To provide additional information to the compiler, for optimization

D. To provide the programmer an alternate control flow option

What is the Result?

```
public class A {
    private String aMethod() {
        return "Private";
    }
}

A myA = new A();
System.out.print(myA.aMethod());
```

A. Won't compile

B. MethodNotFoundException
   throw

C. "Private" printed

D. null printed

What does "Static" do?

A. To indicate a value won't change during the program's execution

B. To indicate a variable will change frequently during the program's execution

C. To indicate that a method or variable is connected to the class, and not instances of the class

D. To indicate that a method or variable is connected to the instance, and not the class itself

What is Printed?

```
public class Parent {
    public String aMethod() {
        return "Parent";
    }
}

public class Child extends Parent {
    public String aMethod() {
        return "Child";
    }
}

Child aChild = new Child();
String result = aChild.aMethod();
System.out.print(result);
```

A. "Parent"

B. "Child"

C. "ChildParent"

D. "ParentChild"

Done!

Housekeeping

- Debugging SSH / git password issues
- SSH keys and ~/.ssh/config
- Abbreviated Office Hours Today
- Homework 2

Homework 2

- Animal
  - Beluga
  - Chameleon
  - Dog
  - GermanShepard
  - Mammal
  - Orca
  - Reptile
  - ShibaInu
  - Snake
  - Whale

1. isWarmBlooded
2. isLivingUnderWater
3. isNamedAfterEuropeanCountry
4. canChangeColor
5. isBlackAndWhite

Class Visibility

Problem

- Classes represent data and functionality
- Some functionality is for "users" of code
- Some functionality is "internal" or "sensitive"

Credit Card Account

Credit Account

Charge Account

Credit Card Account

Credit Account

- Code to verify status of account (ping charge)
- Code to **send** money to account

Charge Account

- Code to verify status of account (ping charge)
- Code to **take** money to account

Credit Card Account

Credit Account

- Call "Verify Account"
- Send money to account

Charge Account

- Call "Verify Account"
- Take money from account

Verify Account

- Code to verify status of account (ping charge)

Credit Card Account

Credit Account

- Call "Verify Account"
- Send money to account

Charge Account

- Call "Verify Account"
- Take money from account

Verify Account

!!!!!!!!

- Code to verify status of account (ping charge)

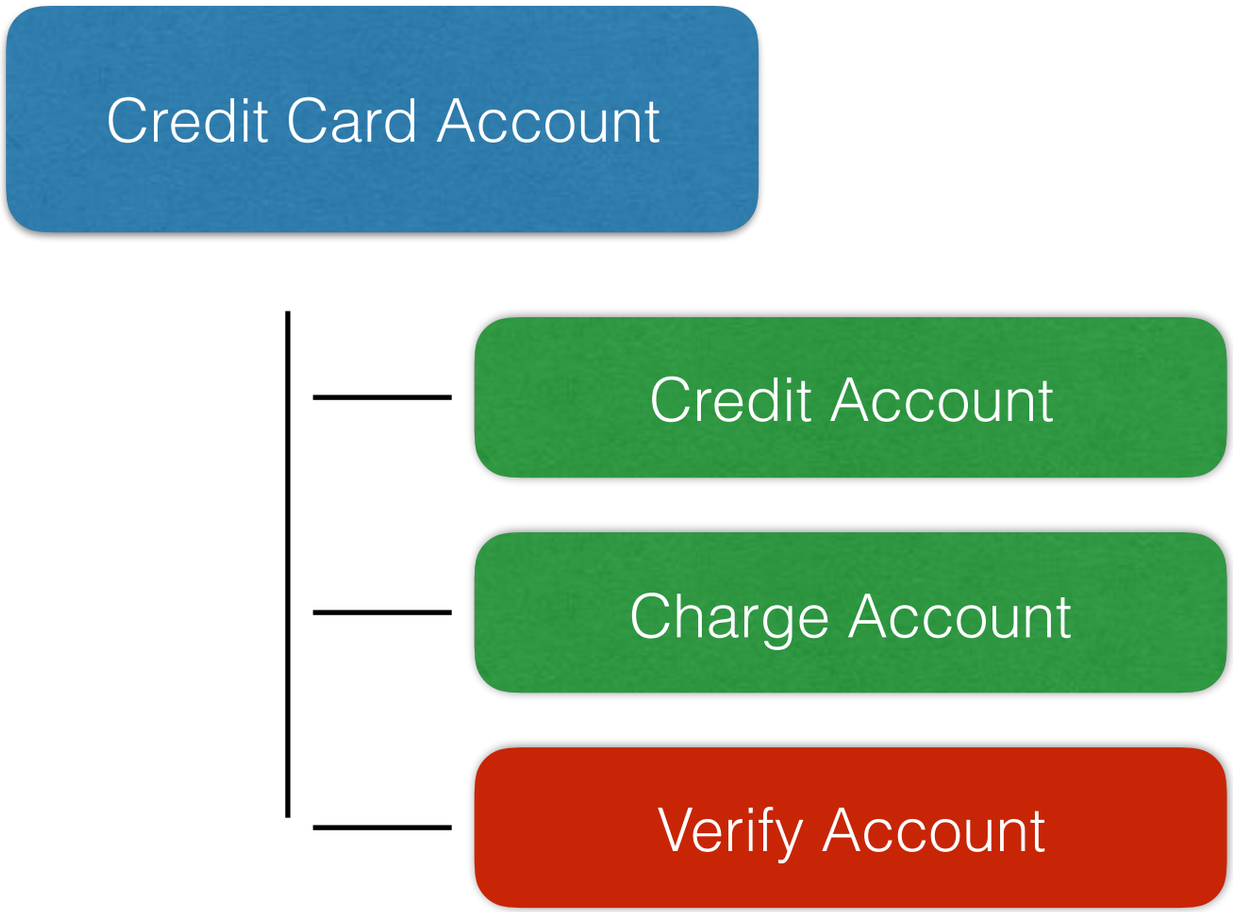
Java's Solution

- **Public**
  Internal and external access

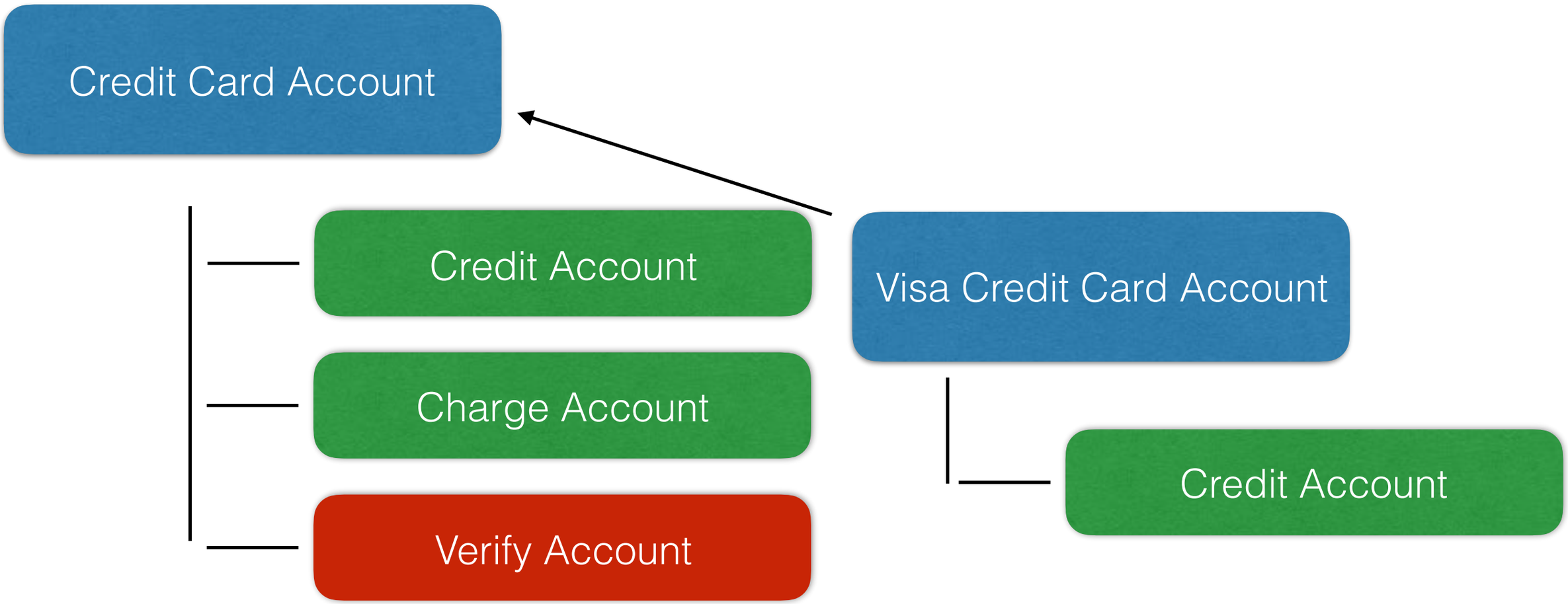
- **Private**
  Internal class access only

- Properties and Methods

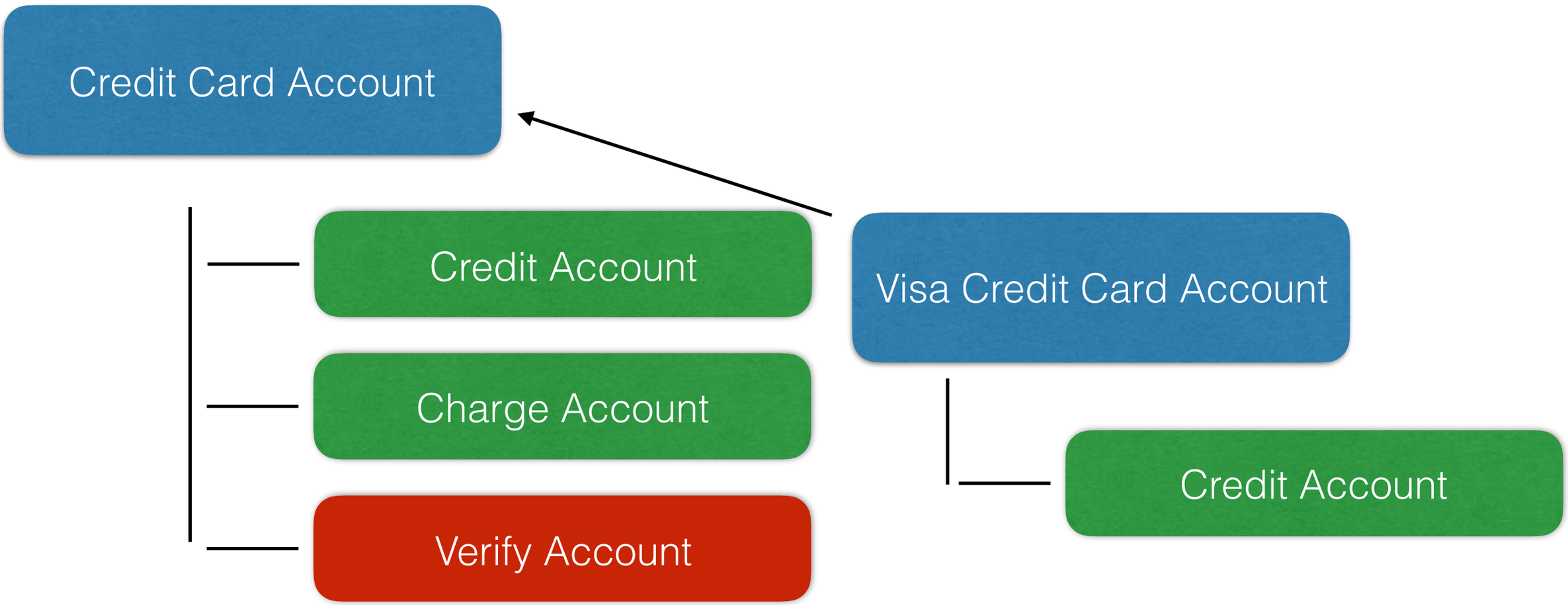
CreditCardAccount.java -->


Credit Card Account
Credit Account
Charge Account
Verify Account

Subclassing



Subclassing



Subclass "Credit Account" cannot see "Verify Account"

Java's Solution

- **Public**
  Internal and external access

- **Private**
  Internal class access

- **Protected**
  Internal class and subclass access

CreditCardAccount.java -->

Discussion

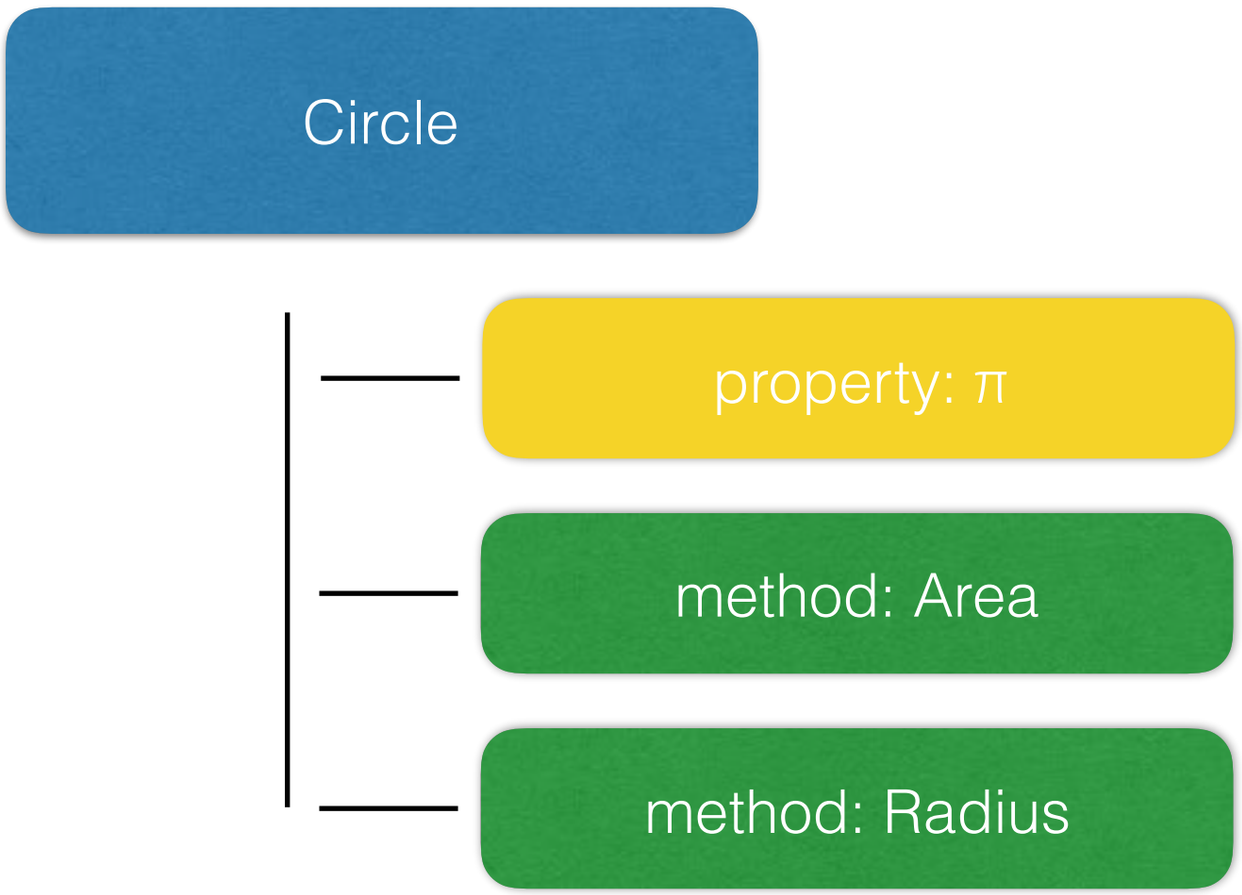
- Other examples for public / private / protected methods or properties
- Security
- Shared Functionality
- When subclasses want to share internal functionality / data

"Static"

Problem

- Instances describe functionality and data of a type
- Where to put code and data shared between instances?
- Constants, aggregators, "Importing", etc…

Constants



Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

Circle

property: π

method: Area

method: Radius

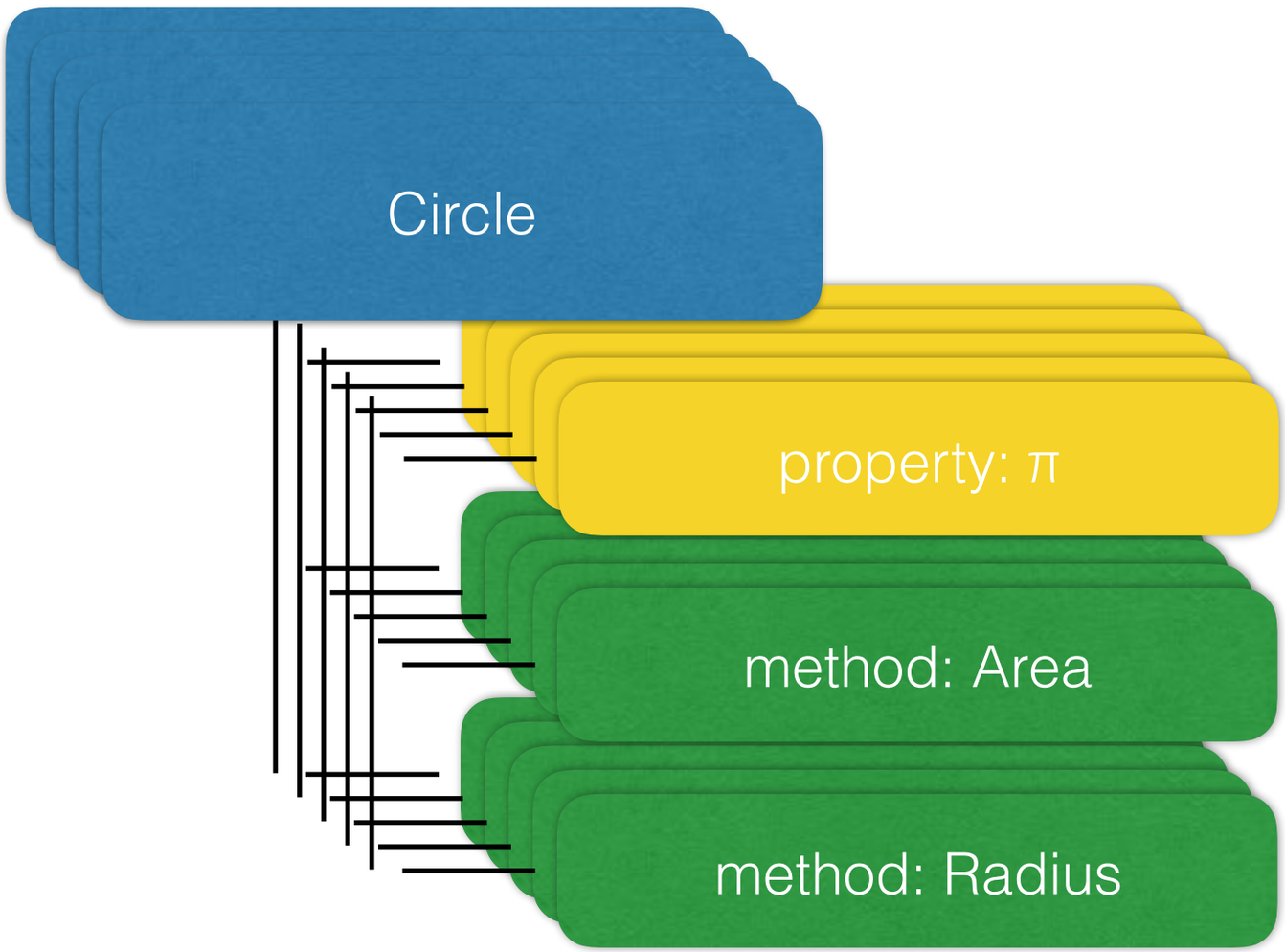
Circle

property: π

method: Area

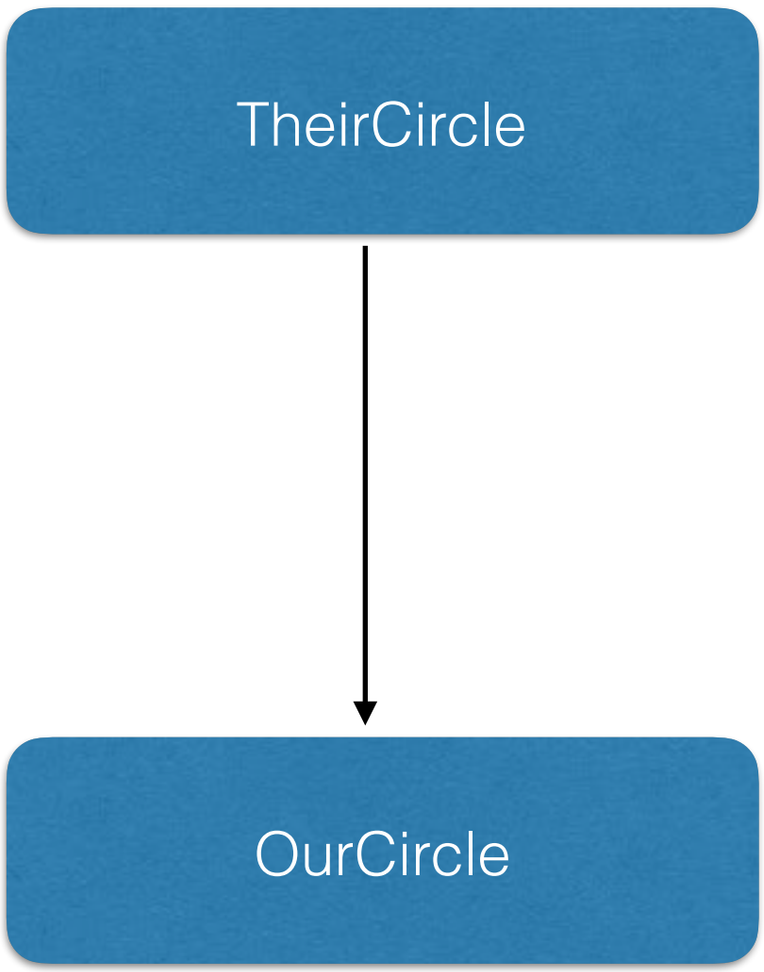
method: Radius

Aggregation



- We want to find the area of multiple circles
- We want to store that somewhere in our code
- Could create a "CircleAdder" class… awkward…

Converters



- We're using someone else's code
- We want to convert their types into our types
- Could create a "CircleConverter" class…
- awkward…

Java's Solution

- **Static:** Attach functionality and data to classes (instead of instances)

- **Can also be public / private / protected**
  Same access rules as with instances

- Properties and Methods

Static vs Instance

	Live on…	Can Access…	Number in system
Instance	Each instance	The fields in the same instance	Arbitrarily many
Static	The class	Only other static values	Just one

Circle.java -->

Exceptions

Problem

- Errors happen, all the time (Sod's Law)
- Robust programs need to deal with these errors
- Some errors are **REALLY** bad
- Programmers are lazy
- Programming languages should nudge in the right direction…

Error Example

- writeStuffToDisk(byte[] lotsOfStuff)
- What to do when things go wrong?

Error Example

- writeStuffToDisk(byte[] lotsOfStuff)
- What to do when things go wrong?
  - **Return boolean**

boolean writeStuffToDisk(byte[] lotsOfStuff)

Error Example

- writeStuffToDisk(byte[] lotsOfStuff)
- What to do when things go wrong?
  - **Return boolean**
    boolean writeStuffToDisk(byte[] lotsOfStuff)
  - **Return "error code"**
    int writeStuffToDisk(byte[] lotsOfStuff)

Error Example

- writeStuffToDisk(byte[] lotsOfStuff)
- What to do when things go wrong?
  - **Return boolean**
    boolean writeStuffToDisk(byte[] lotsOfStuff)
  - **Return "error code"**
    int writeStuffToDisk(byte[] lotsOfStuff)
  - **Multiple values**
    ErrorAndResultObject writeStuffToDisk(byte[] lotsOfStuff)

Error Example

- writeStuffToDisk(byte[] lotsOfStuff)
- What to do when things go wrong?
  - **Return boolean**
    boolean writeStuffToDisk(byte[] lotsOfStuff)
  - **Return "error code"**
    int writeStuffToDisk(byte[] lotsOfStuff)
  - **Multiple values**
    ErrorAndResultObject writeStuffToDisk(byte[] lotsOfStuff)
  - **Out parameter**
    boolean writeStuffToDisk(byte[] lotsOfStuff, Error errorObject)

Problems With these Approaches

?

Problems With these Approaches

- Not enough information
- Large, non-meaningful lookup tables
- Easy to ignore

Java's Solution

- Force programmers to deal with errors through the compiler
- try / catch / finally
  - **try**
    I'm about to do something risky
  - **catch**
    Here's how I'll handle this bad thing that could happen
  - **finally**
    Good or bad, do this last thing
- Part of the method signature

Exceptions Example

```
public FileReader(String fileName) throws FileNotFoundException
```

Exceptions Example

```
public FileReader(String fileName) throws FileNotFoundException
```

```
FileReader secretsReader = new FileReader("secrets.txt");
```

Exceptions Example

```
public FileReader(String fileName) throws FileNotFoundException

FileReader secretsReader = new FileReader("secrets.txt");
```

What could go wrong?

Exceptions Example

```
public FileReader(String fileName) throws FileNotFoundException

try {

    FileReader secretsReader = new FileReader("secrets.txt");

} catch (FileNotFoundException error) {

    System.out.println("Welp, seems we couldn't get the file…");
}
```

Exceptions Example

```
public FileReader(String fileName) throws FileNotFoundException
public int read(char[] cbuf) throws IOException


try {

    FileReader secretsReader = new FileReader("secrets.txt");
    char[] secrets = new char[1000];
    secretsReader.read(secrets);

} catch (FileNotFoundException error) {

    System.out.println("Welp, seems we couldn't get the file…");

} catch (IOException otherError) {
    System.out.println("Got the file, but couldn't read it…");
}
```

URL.java -->

In Groups

- https://www.cs.uic.edu/~psnyder/cs342-summer2017/ic/Student.java

- Walk through the code

- Discuss how this code could be improved using visibility modifiers and static

In Groups

- Code up another example
- Split into two files, Main.java and Student.java
- Implement discussed changes