Introduction to Object-Oriented Design for Scientists

RSE Webinar – August 29th, 2018

BALL STATE
UNIVERSITY

Huseyin Ergin, Ph.D.
Department of Computer Science
http://www.cs.bsu.edu/~hergin

Programming languages

BALL STATE UNIVERSITY

A non-complex program


```
int a = 5
int b = 6
int c = a + b

print c
```

How about longer ones?

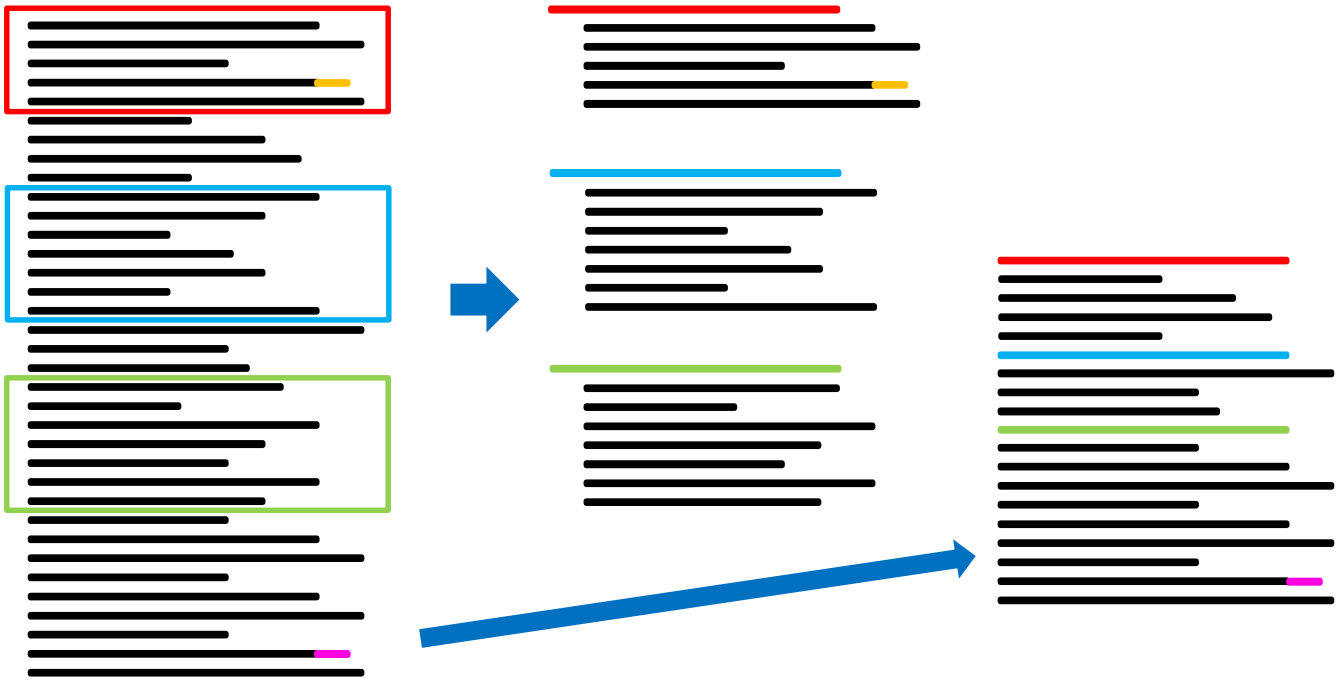
- Chrome browser: 17 millions LOC (lines of code)
- Office 2013: 45 millions LOC
- Facebook: 60 millions LOC

BALL STATE
UNIVERSITY

There are some solutions...

BALL STATE
UNIVERSITY

When things get bigger → Modularize



BALL STATE UNIVERSITY

Modularize even more...

BALL STATE UNIVERSITY

Modularizing


```
1
2   print("I am a dog")
3   print("I have 4 legs")
4   print("My sound is: woof woof")
5
```

```
1
2   def dogInformation():
3       print("I am a dog")
4       print("I have 4 legs")
5       print("My sound is: woof woof")
6
7   dogInformation()
8   dogInformation()
9   dogInformation()
10
```

Modularizing - parametrized


```
1
2   def dogInformation():
3       print("I am a dog")
4       print("I have 4 legs")
5       print("My soun
6
7   dogInformation()
8   dogInformation()
9   dogInformation()
10
```

```
1
2   def dogInformation():
3       print("I am a dog")
4       print("I have 4 legs")
                              y sound is: woof woof")
                              y name is Rexx")
                              ion()
                              ion()
                              ion()
```

```
1
2   def dogInformation(name):
3       print("I am a dog")
4       print("I have 4 legs")
5       print("My sound is: woof woof")
6       print("My name is " + name)
7
8   dogInformation("Rexx")
9   dogInformation("Bingo")
10  dogInformation("Rocky")
11
```

BALL STATE UNIVERSITY

Modularizing – more parametrized


```
1
2   def chickenInformation(name):
3       print("I am a chicken")
4       print("I have 2 legs")
5       print("My sound is: cluck cluck")
6       print("My name is " + name)
7
8   chickenInformation("Angel")
9   chickenInformation("Coco")
10
11  def dogInformation(name):
12      print("I am a dog")
13      print("I have 4 legs")
14      print("My sound is: woof woof")
15      print("My name is " + name)
16
17  dogInformation("Rexx")
18  dogInformation("Bingo")
19  dogInformation("Rocky")
20
```

```
1
2   def animalInformation(type, legCount, sound, name):
3       print("I am a " + type)
4       print("I have " + legCount + " legs")
5       print("My sound is: " + sound)
6       print("My name is " + name)
7
8   animalInformation("chicken", 2, "cluck cluck", "Angel")
9   animalInformation("chicken", 2, "cluck cluck", "Coco")
10
11  animalInformation("dog", 4, "woof woof", "Rexx")
12  animalInformation("dog", 4, "woof woof", "Bingo")
13  animalInformation("dog", 4, "woof woof", "Rocky")
14
```

BALL STATE UNIVERSITY

Epiphany

- Wouldn't it be nice if there is special structure that:
  - Knows what kind of an animal it is.
  - Knows how many legs it has.
  - Can make sound by itself.
  - Has a name.

Epiphany

BALL STATE UNIVERSITY

In object-oriented design, we are copying the real life structure in our application.

“

First, we want to establish the idea that a computer language is not just a way of getting a computer to perform operations but rather it is a novel formal medium for expressing ideas about methodology.

Thus, programs must be written for people to read, and only incidentally for machines to execute.

”

Harold Abelson

In his book SICP: Structure and Interpretation of Computer Programs (with Jay Sussman)

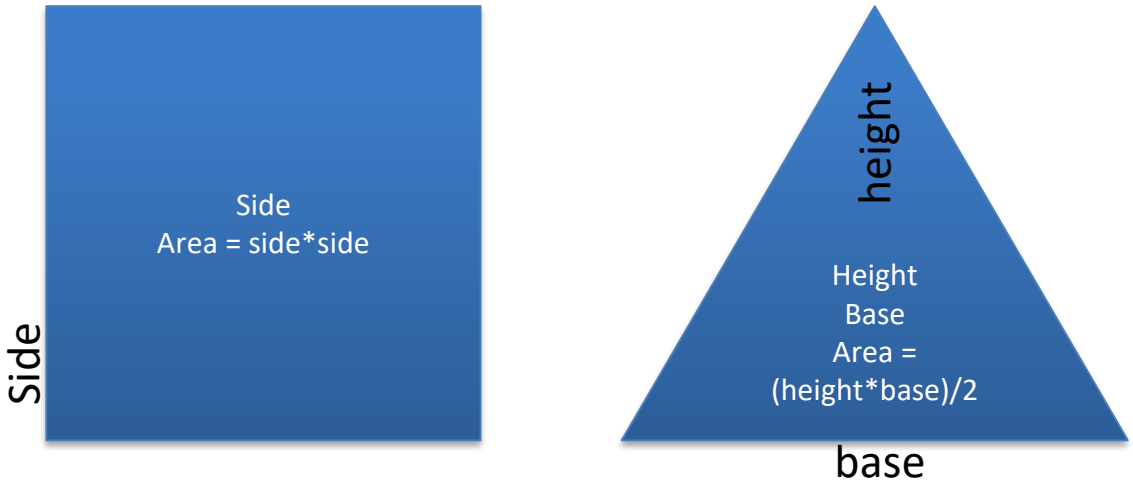
BALL STATE
UNIVERSITY

Epiphany

- Wouldn't it be nice if there is special structure that:
  - Knows what kind of an animal it is.
  - Knows how many legs it has.
  - Can make sound by itself.
  - Has a name.

OBJECT-ORIENTED DESIGN

Object-oriented design



Object identification

- Car, glass, bike, bus
- Dog, cat, person, laptop
- Running, walking, making a sound etc… ?
- Side, height, numberOfLegs etc… ?

Everything can be an object…

- It just depends on the context!
- Distance between two cities can be an object

- Relationship between two people can be an object

Object-oriented languages

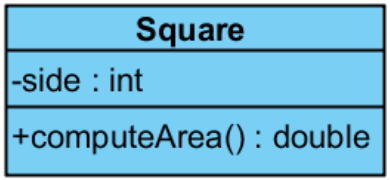
- Most modern languages support object-orientation now
  - Java
  - C++ (C with classes)
  - C#
  - Python
  - Even Fortran
  - Any many more

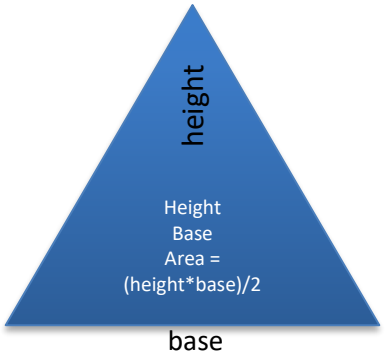
Now, various languages support object-orientation, there should be a way to represent this system regardless of the language we use.

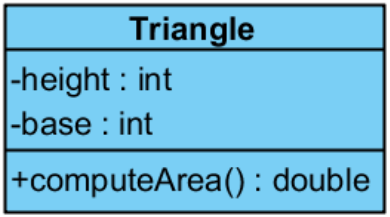
BALL STATE
UNIVERSITY


UNIFIED
MODELING
LANGUAGE
UML
TM

UML basics

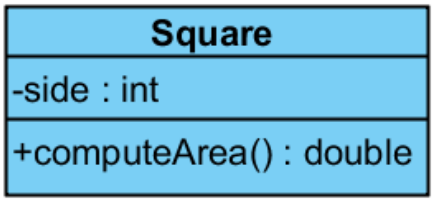


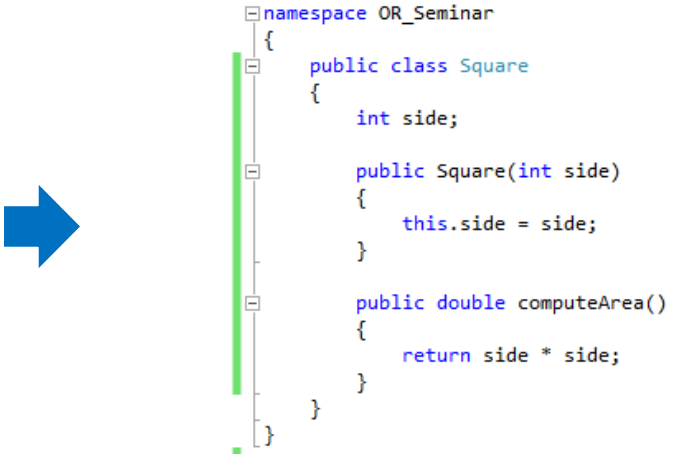




BALL STATE
UNIVERSITY

Class in C#



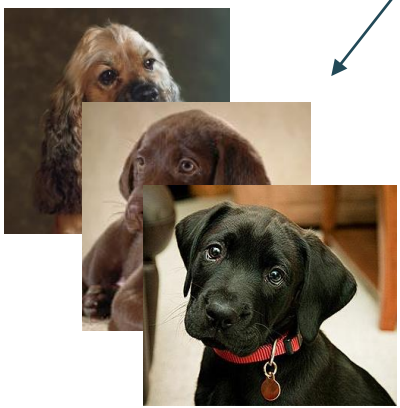


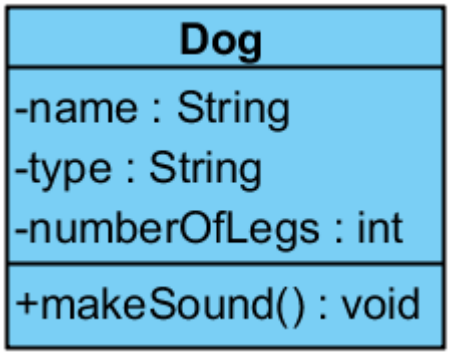
```
namespace OR_Seminar
{
    public class Square
    {
        int side;

        public Square(int side)
        {
            this.side = side;
        }

        public double computeArea()
        {
            return side * side;
        }
    }
}
```

BALL STATE UNIVERSITY

WAIT!

- We were talking about objects, now what the heck is a class?

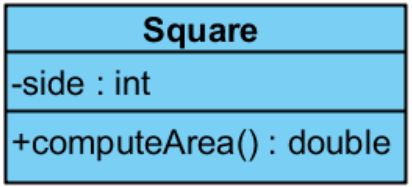




BALL STATE UNIVERSITY

Objects from classes

- Constructor




```
namespace OR_Seminar
{
    public class Square
    {
        int side;

        public Square(int side)
        {
            this.side = side;
        }

        public double computeArea()
        {
            return side * side;
        }
    }
}
```

- Now, we can create objects

```
namespace OR_Seminar
{
    class Program
    {
        static void Main(string[] args)
        {
            Square square1 = new Square(5);
            Square another = new Square(10);
        }
    }
}
```

BALL STATE UNIVERSITY

How it affects our code?


```
1
2   def animalInformation(type, legCount, sound, name):
3       print("I am a " + type)
4       print("I have " + legCount + " legs")
5       print("My sound is: " + sound)
6       print("My name is " + name)
7
8   animalInformation("chicken", 2, "cluck cluck", "Angel")
9   animalInformation("chicken", 2, "cluck cluck", "Coco")
10
11  animalInformation("dog", 4, "woof woof", "Rexx")
12  animalInformation("dog", 4, "woof woof", "Bingo")
13  animalInformation("dog", 4, "woof woof", "Rocky")
14
```

```
1   class Animal:
2       def __init__(self,typ,leg_count,sound,name):
3           self.type = typ
4           self.leg_count = leg_count
5           self.sound = sound
6           self.name = name
7
8       def info(self):
9           print("I am a " + self.type)
10          print("I have " + str(self.leg_count) + " legs")
11          print("My sound is: " + self.sound)
12          print("My name is: " + self.name)
13
14
15  class Dog(Animal):
16      def __init__(self, name):
17          Animal.__init__(self, "dog", 4, "woof woof", name)
18
19
20  class Chicken(Animal):
21      def __init__(self, name):
22          Animal.__init__(self, "chicken", 2, "cluck cluck", name)


25  rexx = Dog("Rexx")
26  rexx.info()
27
28  angel = Chicken("Angel")
29  angel.info()
```

BALL STATE UNIVERSITY

Concepts and design principles in OOD

- Concepts:
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Cohesion
  - Coupling
- Principles
  - Open-closed principle
  - Single Responsibility principle
  - Liskov substitution principle
  - Interface segregation principle
  - Dependency inversion principle

Encapsulation

Protecting your information from being used incorrectly

BALL STATE UNIVERSITY

Encapsulation - example


```
1
2  public class Airplane {
3      public int speed;
4
5      public Airplane() { }
6  }
7
```

PROBLEM?

```
1
2  public class Main {
3
4      public static void main(String[] args) {
5          Airplane plane1 = new Airplane();
6          plane1.speed = 100;
7          System.out.println(plane1.speed);
8      }
9
10 }
11
```

BALL STATE
UNIVERSITY

Encapsulation - example


```
1
2  public class Airplane {
3      public int speed;
4
5      public Airplane() { }
6  }
7
```

```
1
2  public class Airplane {
3      private int speed;
4
5      public Airplane() { }
6
7      public void setSpeed(int newSpeed) {
8          this.speed = newSpeed;
9          this.speed = this.speed - 9; // adjustment for rainy weather
10     }
11 }
12
```

SOLUTION?

BALL STATE UNIVERSITY

Encapsulation – relevant example


```
1   public class Exporter {
2
3       private string fileDir = "";
4
5       public void SetFilePath(string newFileDir) {
6           if(FileUtils.DirExists(newFileDir)) {
7               this.fileDir = newFileDir;
8           }
9           throw new FilePathInvalidException("The new file path doesn't exist!");
10      }
11
12  }
```

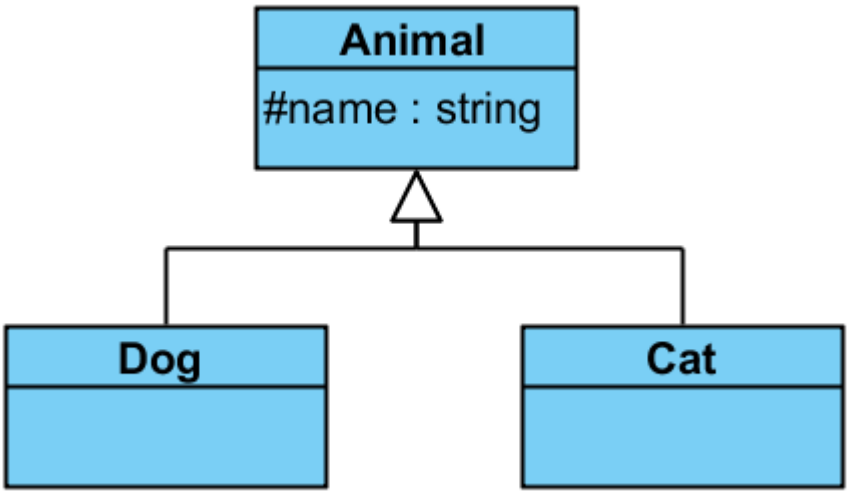
BALL STATE
UNIVERSITY

Inheritance

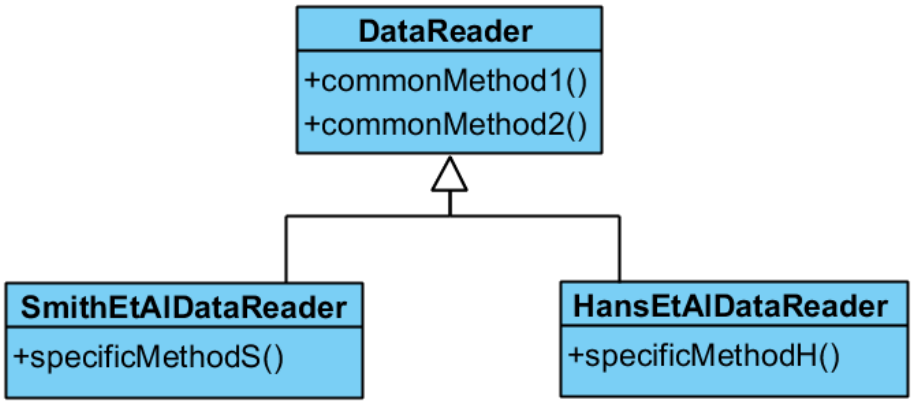
A class can inherit properties and operations from another class.

BALL STATE
UNIVERSITY

Inheritance - example



Inheritance – relevant example

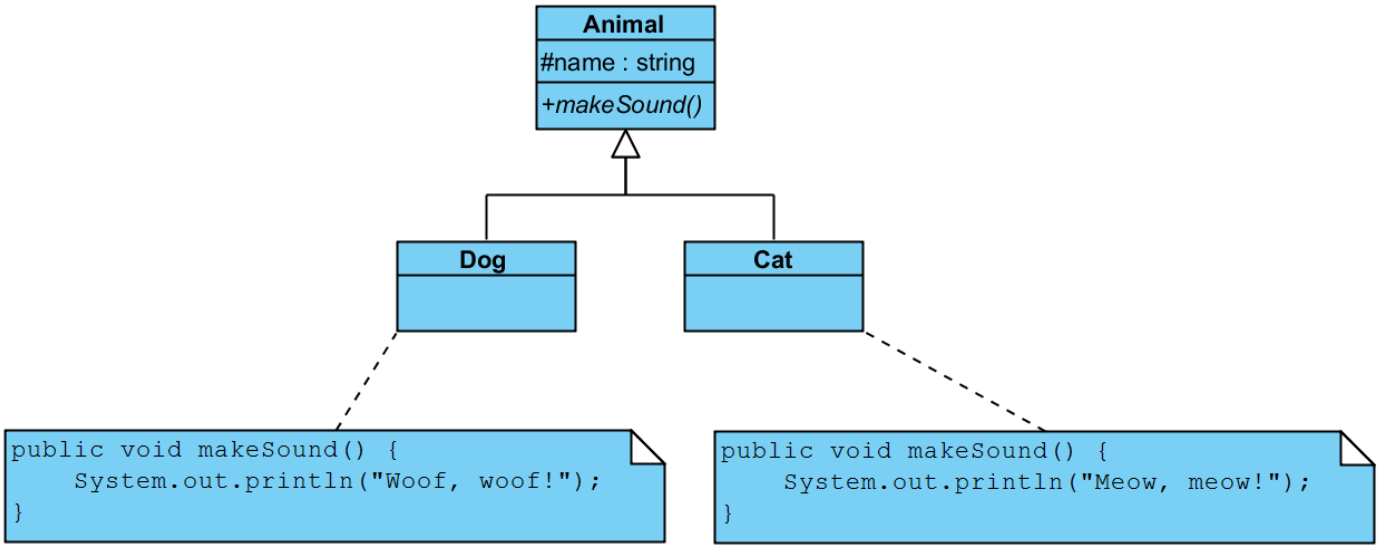


Polymorphism

Having different forms of same operations.

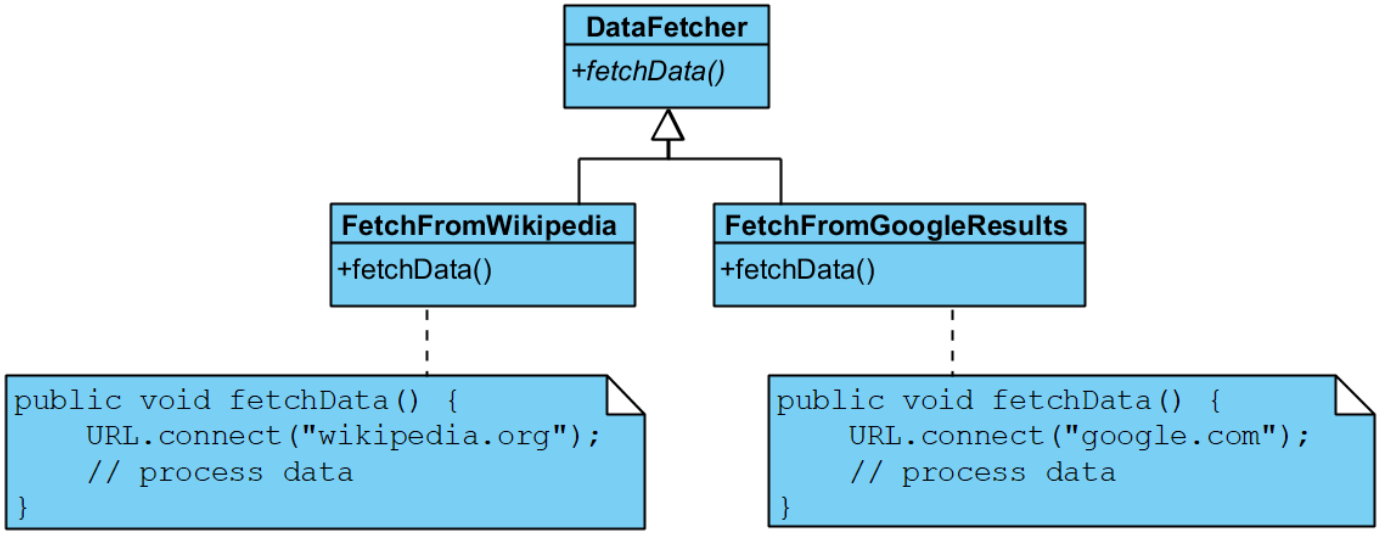
BALL STATE
UNIVERSITY

Polymorphism - example



BALL STATE UNIVERSITY

Polymorphism – relevant example



BALL STATE UNIVERSITY

Cohesion

A class should do one thing really well and should not try to do or be something else.

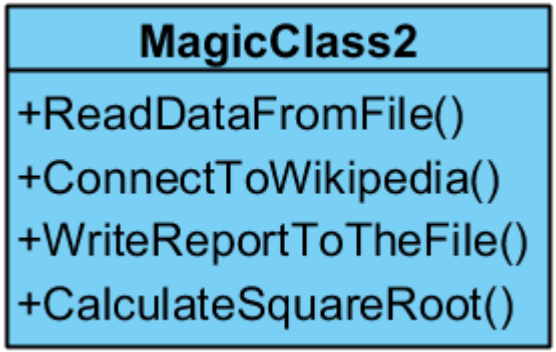
BALL STATE
UNIVERSITY

Cohesion - example


```
public class Magic
{
    public void PrintDocument(Document d) { … }
    public void SendEmail(string recipient,
        string subject, string text) { … }
    public void CalculateDistanceBetweenPoints(
        int x1, int y1, int x2, int y2) { … }
}
```

BALL STATE
UNIVERSITY

Cohesion – relevant example



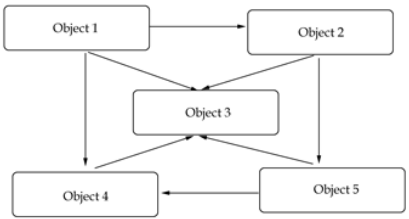
Coupling

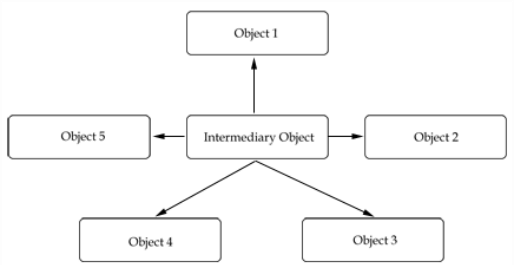
The extent to which classes depend on one another.

A class should work independently without being coupled too much to other classes, which help making them modules and available on demand.

BALL STATE
UNIVERSITY

Coupling - overview





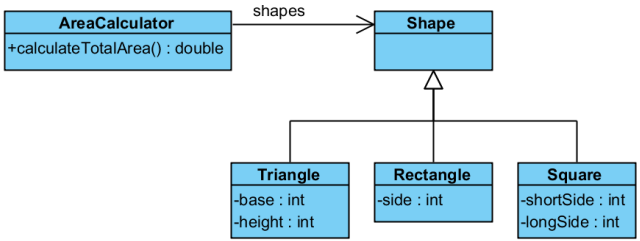
BALL STATE UNIVERSITY

Open-Closed Principle

Classes should be open to extension, but closed for modification.

BALL STATE
UNIVERSITY

Open-closed principle - example

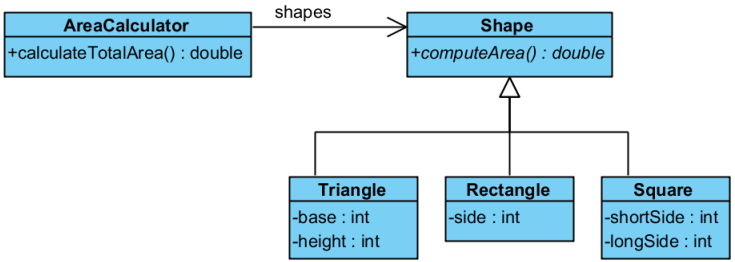


PROBLEM?

```
public double calculateTotalArea() {
    double result = 0;

    // VIOLATES the OCP principle
    for (Shape shape : shapes) {
        if (shape instanceof Triangle) {
            result += ((Triangle) shape).base * ((Triangle) shape).height / 2;
        } else if (shape instanceof Square) {
            result += ((Square) shape).length * ((Square) shape).length;
        } else if (shape instanceof Rectangle) {
            result += ((Square) shape).length * ((Square) shape).length;
        }
    }

    return result;
}
```

BALL STATE UNIVERSITY

Open-closed principle - example



SOLUTION?

```java
public double calculateTotalArea() {
    double result = 0;

    // VIOLATES the OCP principle
    for (Shape shape : shapes) {
        if (shape instanceof Triangle) {
            result += ((Triangle) shape).base * ((Triangle) shape).height / 2;
        } else if (shape instanceof Square) {
            result += ((Square) shape).length * ((Square) shape).length;
        } else if (shape instanceof Rectangle) {
            result += ((Square) shape).length * ((Square) shape).length;
        }
    }

    return result;
}
```

```java
34      public double calculateTotalArea() {
35          double result = 0;
36
37          // Fixing the OCP principle violation
38          for (Shape shape : shapes) {
39              result += shape.computeArea();
40          }
41
42          return result;
43      }
```

BALL STATE UNIVERSITY

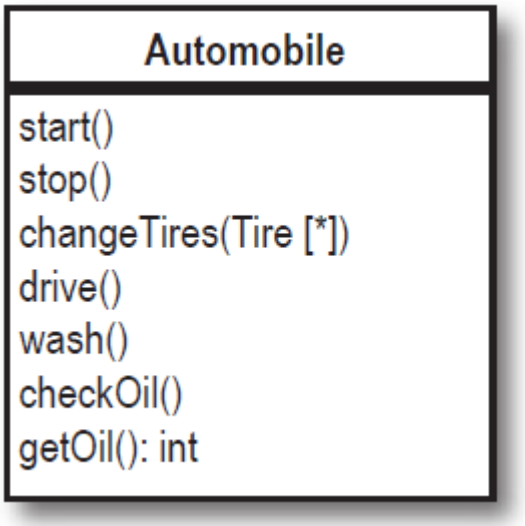
Single Responsibility Principle

Every object in the system should have one responsibility. Therefore, one reason to change.

BALL STATE
UNIVERSITY

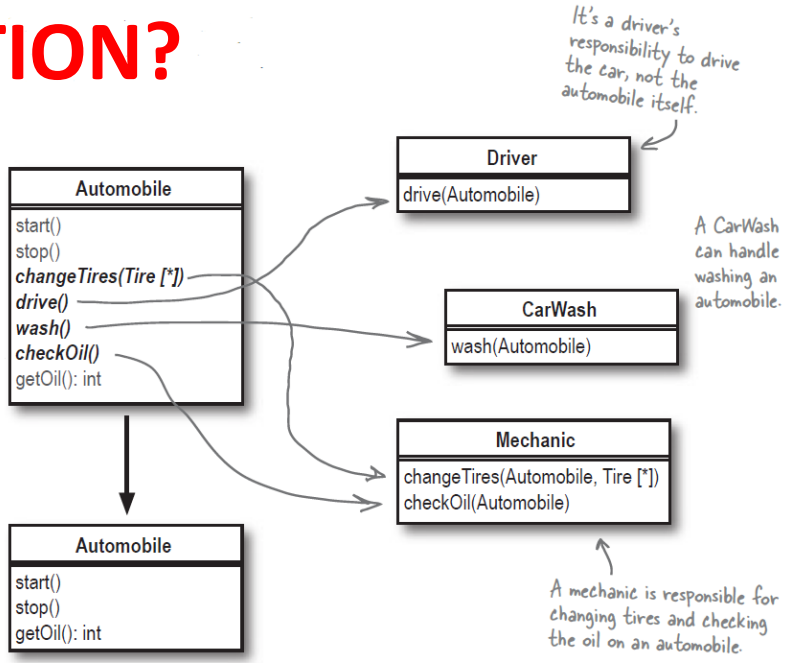
Single responsibility principle - example

PROBLEM?



Single responsibility principle - example

SOLUTION?



BALL STATE UNIVERSITY

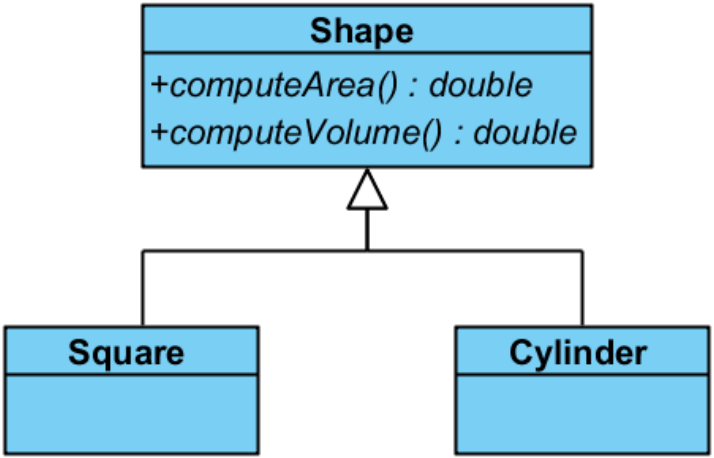
Interface Segregation Principle

A class should never be forced to have some unnecessary methods.

BALL STATE UNIVERSITY

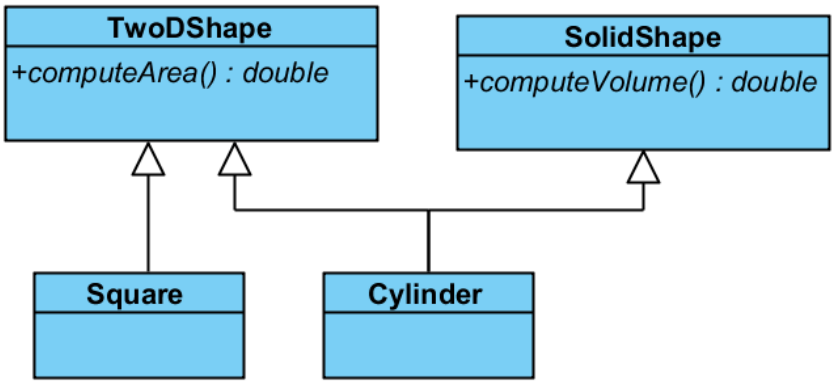
Interface segregation principle - example

PROBLEM?



Interface segregation principle - example

SOLUTION?



So, what can we achieve from this? – Sample scenarios

- Read data from various sources but our program can stay the same
- Easily switch between different algorithms on the same data
- Make any module work independent from others
- Make the output of the project independent from the data or the algorithm itself
- Test the correctness of each class independently
- Model the problem in a human-readable way
- And most importantly, reuse and maintain your application better.
- Future-proof.

BALL STATE
UNIVERSITY

Conclusion

- Object-oriented programming provides very flexible structures for our programs.
  - It can be applied in many languages, as long as the language supports object-orientation.
- If we obey the principles, it will be an actual system.
  - Otherwise, it is just the same code with classes and additional complexity.
- Object-oriented system is not a perfect system and it has its own flaws. But it is still the best system.

Always strive for the best design.

BALL STATE
UNIVERSITY

Thanks & Questions

hergin@bsu.edu
http://www.cs.bsu.edu/~hergin

BALL STATE
UNIVERSITY

Some resources

- Object-oriented programming with C# (The book itself is nice and free, chapter 20 is OOP): http://www.introprogramming.info/english-intro-csharp-book/read-online/
- For new starters to OOP, this book is fun: https://www.amazon.com/Head-First-Object-Oriented-Analysis-Design/dp/0596008678
- Detailed explanation, nicely done, 2 pages (Java):
  - https://www.ntu.edu.sg/home/ehchua/programming/java/J3a_OOPBasics.html
  - https://www.ntu.edu.sg/home/ehchua/programming/java/J3b_OOPInheritancePolymorphism.html
- Same as above but with C++:
  - https://www.ntu.edu.sg/home/ehchua/programming/cpp/cp3_OOP.html
- Even though, there are a lot of resources. I suggest to work with someone who you can ask questions immediately. Because OOP requires a change of mindset.

BALL STATE
UNIVERSITY