Lecture: Object-Oriented Programming Features

ESS201: Programming II

Jaya Sreevalsan Nair, IIIT-Bangalore

August 12, 2019

Features of OOP
(Object-Oriented Programming)

Shift from Procedural to OOP

C is a procedural programming language, so is Python (largely).

Procedural programming is a type of imperative programming.

- ▷ Tasks are done iteratively and in functions, as required.
- ▷ It involves iteration, sequencing, selection, and modularization.

What’s missing? Mechanism for **code reuse**.

What happens owing to inability to perform code reuse? Spaghetti code!

A programming paradigm that supports code reuse is Object-oriented programming.

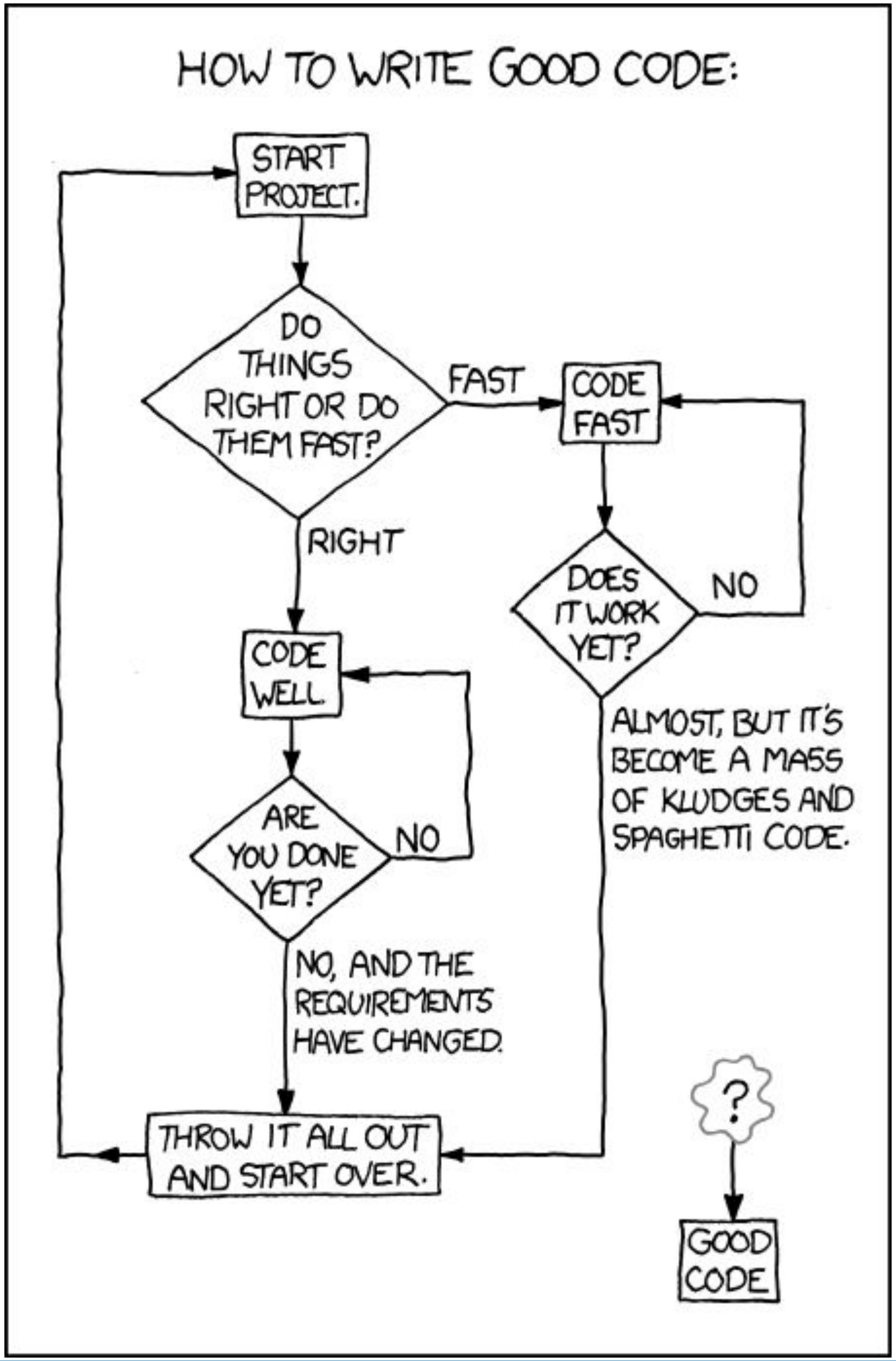
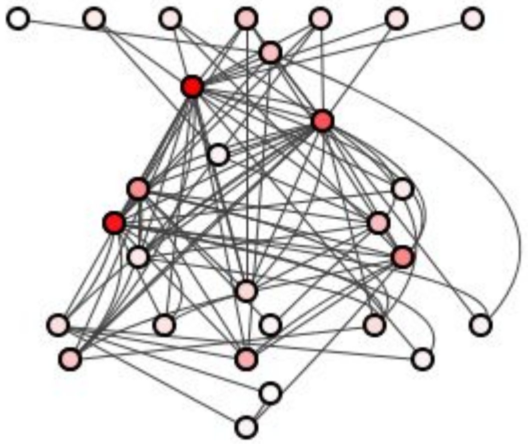
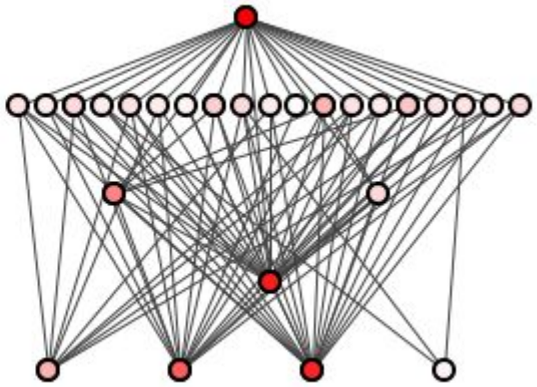

HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

Image courtesy: https://xkcd.com/844/

Structural Disorder in Code

JUnit disorder is 76%, but Spoiklin Soice disorder is 3%. (This however shows dependency graph of packages -- which is a higher level version of code.)





Image+Content courtesy: https://dzone.com/articles/are-we-still-writing-spaghetti-code

Features of OOP

We have classes (grouping of data in a generic fashion) and its instantiations called objects.

What does OOP have to offer?

The key features of OOP enable **code reuse, flexibility in coding, and clean code.**

Its key features are:

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

Encapsulation

1. Encapsulation

Group properties (fields) and behaviour (methods) together in a class, which are inherently present in all objects belonging to the class.

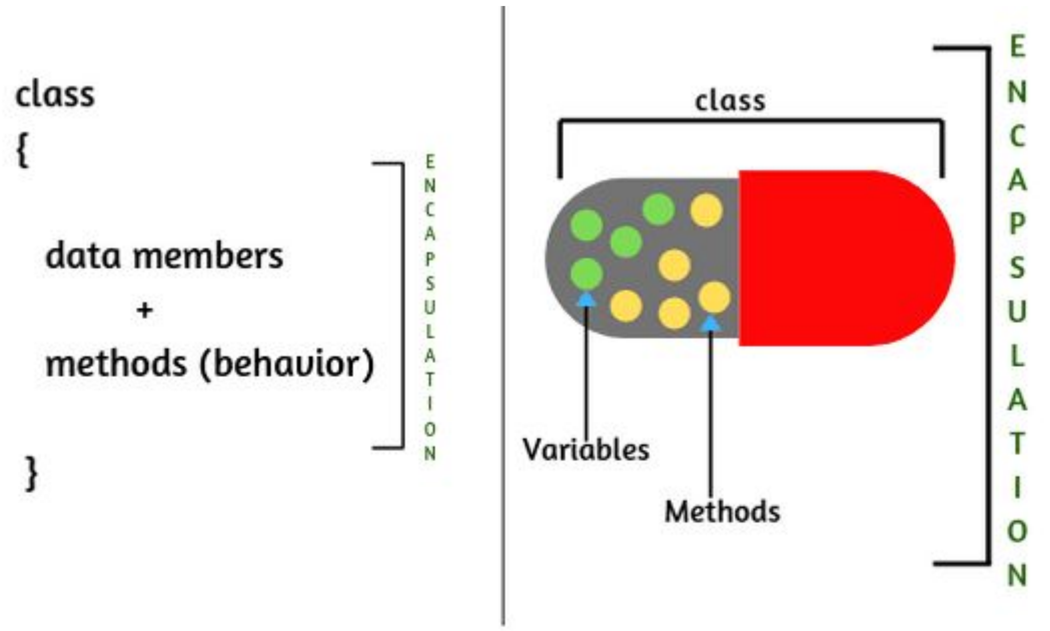


Fig: Encapsulation

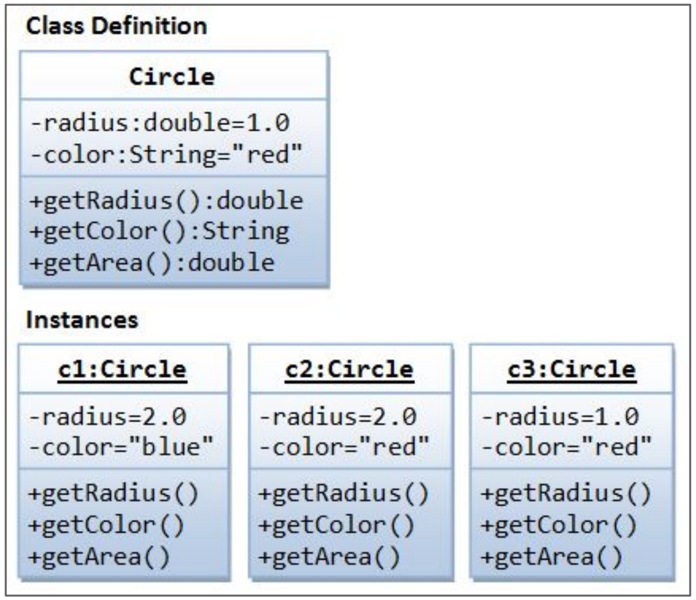
Image courtesy: https://www.scientecheasy.com/2018/06/encapsulation-in-java-real-time-examples-advantages.html

How does encapsulation help?

Think of functions with as few input parameters.

Functions should have a small number of arguments. No argument is best, followed by one, two, and three. More than three is very questionable and should be avoided with prejudice.

- Uncle Bob Martin, “Clean Code”



Instead of getArea(double radius) for any generic circle; now with encapsulation, “Circle” can “own” its property of “area” and can very well compute it any time its “radius” changes.

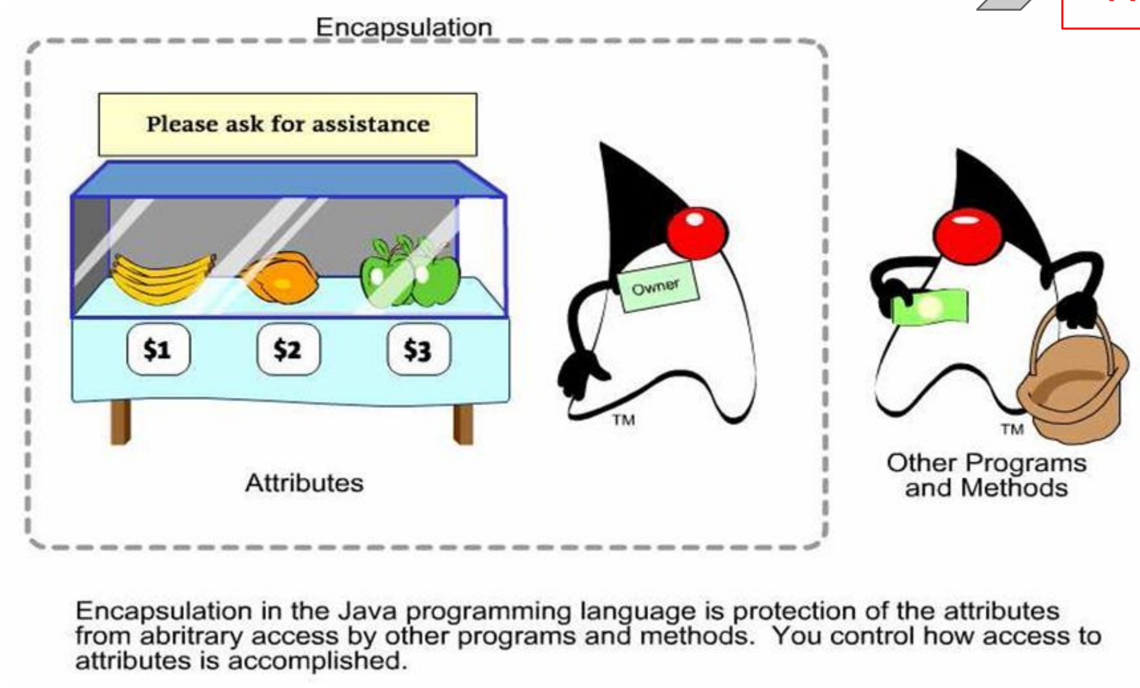
Image courtesy: https://www.ntu.edu.sg/home/ehchua/programming/cpp/cp3_OOP.html

How does encapsulation help?

Think of hiding data by limiting/controlling access of anyone and everyone to the properties of the class.

Encapsulation implements Abstraction.

Applicable for C++ too.



Encapsulation in the Java programming language is protection of the attributes from abritrary access by other programs and methods. You control how access to attributes is accomplished.

Image courtesy: https://ui-ex.com/explore/encapsulation-clipart-java-encapsulation/

Abstraction

2. Abstraction

While encapsulation allows to “hide” data, abstraction is for enabling users to see *only* the relevant details of a class.

```
// without abstraction //

double radius = 4.0 ;

int main() {
   double area = 3.141*radius*radius ;
   std :: cout << area << std :: endl ;
}
```

```
// with abstraction //

class Circle {
  // constructor //
   Circle(double val): radius(val) {}

   public:
     double area(void) {
        return (3.141*radius*radius) ;
      }
   private:
      double radius ;
} ;

int main() {
   Circle c(4.0) ;
   std :: cout << c.area() << std :: endl ;
}
```

How does abstraction help?

Thinking of hiding complexity and offering "simpler" interfaces of the objects to the user.

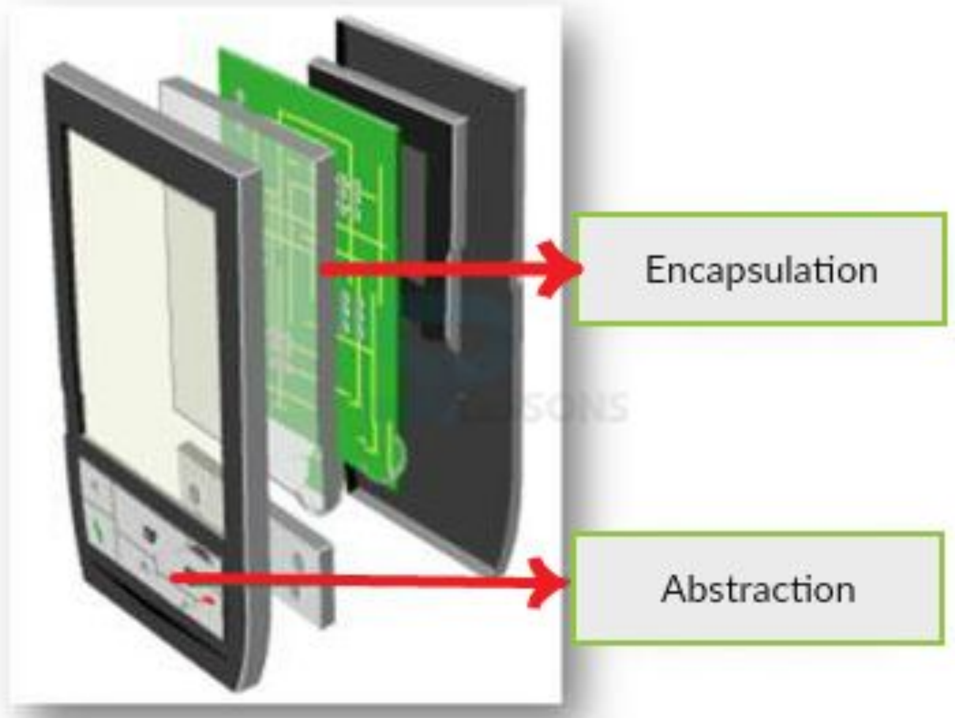


Image courtesy: https://www.splessons.com/lesson/abstraction-in-cpp/

Inheritance

3. Inheritance

Inheritance organizes classes to be a “type” of another class, i.e. a “parent” class, so that several properties and methods can be abstracted commonly across “children” of the class.

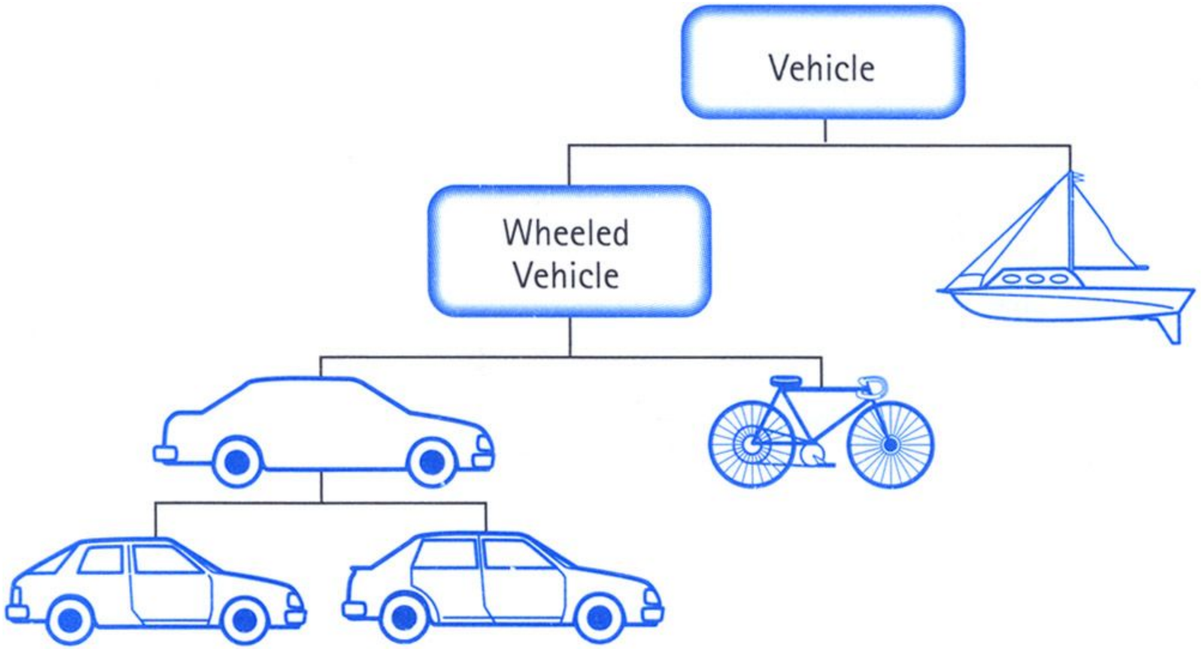


Image courtesy: https://nshahpazov.github.io/new-levels-of-abstractions-with-angularjs-and-es6-classes/

How does inheritance help?

Think of reducing redundant code!

```
class Triangle {
 // 3 vertices
 // method to get area using the 3 vertices
};

class RightTriangle {
  // 3 vertices
  // method to get area using 3 vertices
  // combine right triangles to get an isosceles triangle
};

class EquilateralTriangle {
   // 3 vertices
   // method to get area using 3 vertices
   // combine equilateral triangles to get hexagon
}
```

```
class Triangle {
 // 3 vertices
 // method to get area using the 3 vertices
};

class RightTriangle: public Triangle {
  // combine right triangles to get an isosceles triangle
};

class EquilateralTriangle: public Triangle {
   // combine equilateral triangles to get hexagon
}
```

How does inheritance help?

Inheritance implements Polymorphism

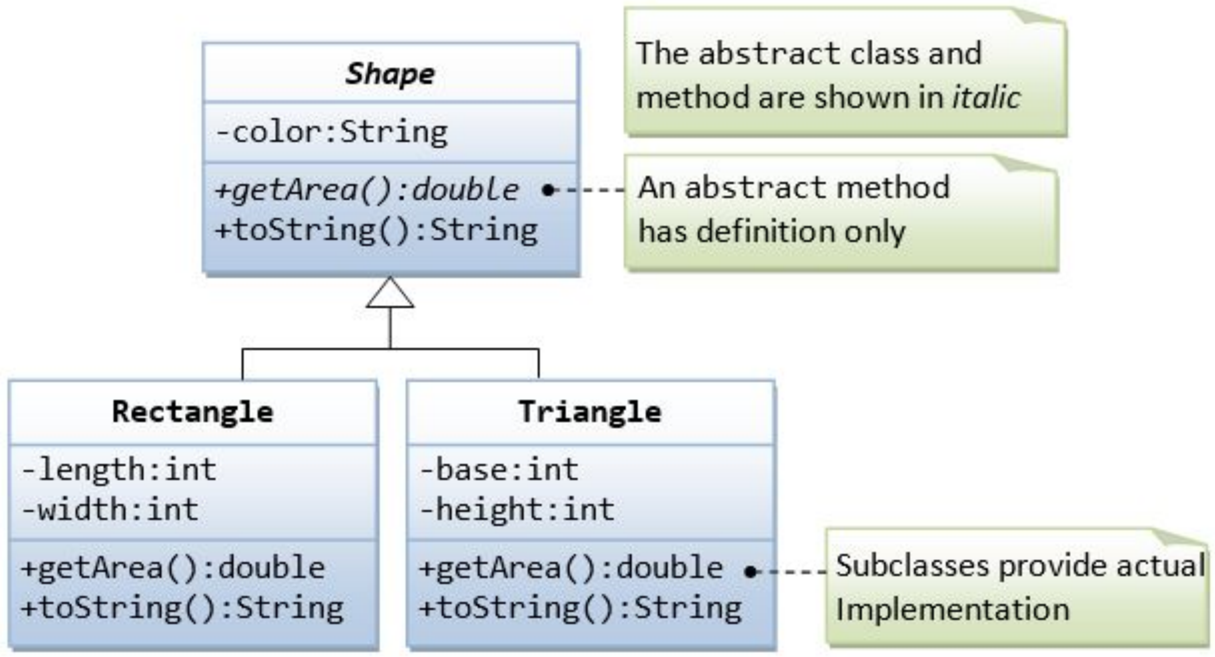


Image courtesy: https://www.ntu.edu.sg/home/ehchua/programming/java/J3b_OOPInheritancePolymorphism.html

Polymorphism

4. Polymorphism

Polymorphism means “many forms”. It allows a method from a base class to behave differently depending on which of its “child” or derived class is calling it.

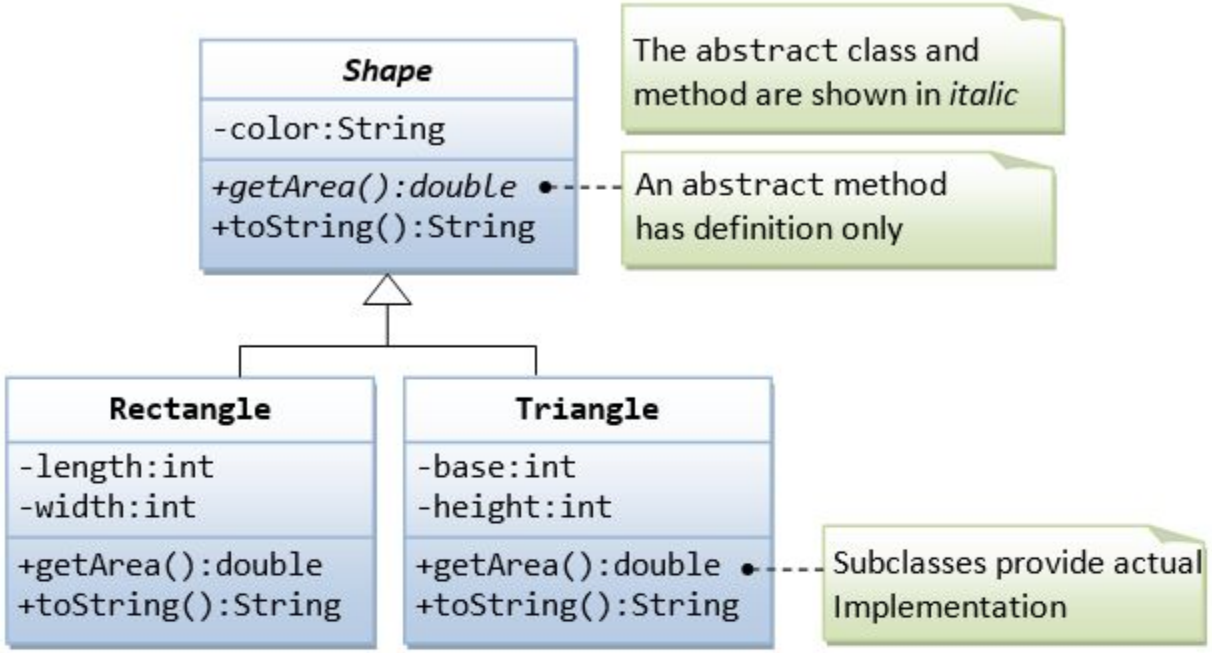


Image courtesy: https://www.ntu.edu.sg/home/ehchua/programming/java/J3b_OOPInheritancePolymorphism.html

How does polymorphism help?

Think of removing redundant switch case statements! The behaviour of the methods "resides" with the concerned class.

```
class Shape {
       int derivedClassID ;
       double area(void) {
           switch (derivedClassID) {
               case 'rectangle': .... ;
               case 'triangle': ..... ;
           } ;
        }
} ;

int main()  {
  Triangle element ;
  std :: cout <<  element.area() ;
}
```

```
class Shape {
      double area(void) {} = 0 ;
} ;
class Rectangle {
       double area(void) { ... }
} ;
class Triangle {
       double area(void) { ... }
} ;

int main()  {
  Triangle element ;
  std :: cout <<  element.area() ;
}
```

While "static binding" is done routinely in inheritance, polymorphism allows for "dynamic binding". The former is done at compile-time, and latter at run-time.

Summary: Features of OOP

Encapsulation: Reduce complexity + increase reusability

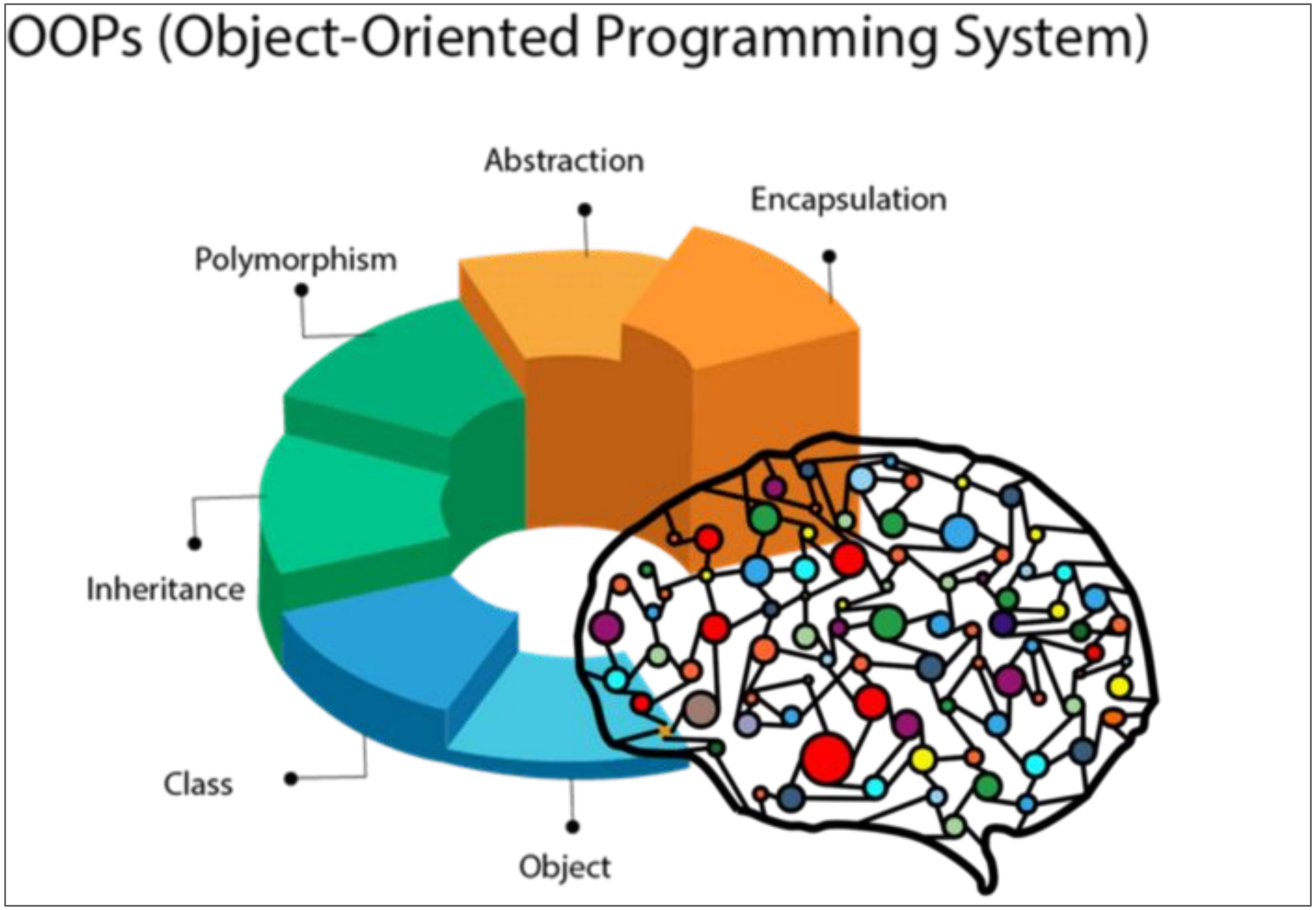
Abstraction: Reduce complexity + isolate impact of changes

Inheritance: Eliminate redundancy

Polymorphism: Refactor unnecessary inflexible switch/case statements

Courtesy:https://www.youtube.com/watch?v=pTB0EiLXUC8

OOP is of great value to data science!



https://towardsdatascience.com/how-a-simple-mix-of-object-oriented-programming-can-sharpen-your-deep-learning-prototype-19893bd969bd

SOLID Principles

SOLID is an acronym which stands for the five basic principles which help to create good software architecture.

- ▷ **S** stands for **SRP** (Single responsibility principle)
- ▷ **O** stands for **OCP** (Open closed principle)
- ▷ **L** stands for **LSP** (Liskov substitution principle)
- ▷ **I** stand for **ISP** (Interface segregation principle)
- ▷ **D** stands for **DIP** (Dependency inversion principle)