Object Oriented Programming Concepts

Supriya Vadiraj

University of Applied Sciences Bonn-Rhein-Sieg

September 12, 2018

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Introduction

- Writing object-oriented programs involves creating classes.
- Creating objects from those classes.
- Creating applications, which are stand-alone executable programs that use those objects.

Features in OOPS

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

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Benifits of OOPS

- Code reuse - programmer efficiency
- Encapsulation -code quality, ease of maintenance
- Inheritance - efficiency, extensibility.
- Polymorphism - Robustness of code

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Object Oriented Principles - Class and Objects

- A class is a user defined blueprint or prototype from which objects are created.
- Object : It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods.

Breed: Bulldog
Size: large
Colour: light gray
Age: 5 years

Dog1Object

Breed: Beagle
Size: large
Colour: orange
Age: 6 years

Dog2Object

Breed: German Shepherd
Size: large
Colour: white & orange
Age: 6 years

Dog3Object

Dog

Fields
Breed
Size
Colour
Age

Methods
Eat()
Run()
Sleep()
Name()

Object Oriented Principles -Inheritance

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the help of inheritance information are managed in a hierarchical manner.
Example:

```
public class Parent {

        public Parent()
            {

        }
}
```

```
public class Child
    extends Parent {

        public Child() {
                super();
        }

}
```

What does that mean? Among other:

- Child inherits all properties and skills.
- Redundancies can be avoided.

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Advantages of Inheritance

- **Reusability**: Inheritance helps the code to be reused in many situations.
- Using the concept of inheritance the programmer can create as many derived classes from the base class as needed while adding new and specific features to the derived class as needed.
- Also the structure of the program is maintained reducing time and effort of the programmer.

Object Oriented Principles -Polymorphism

- Polymorphism in allows subclasses of a class to define their own unique behaviors and yet share some of the same functionality of the parent class.
  - compile-time polymorphism.
  - runtime polymorphism.
- The classic example is the Shape class and all the classes that can inherit from it (square, circle, dodecahedron, irregular polygon, splat and so on).

```
class Car:
    def __init__(self, name):
        self.name = name

    def drive(self):
        raise NotImplementedError("Subclass must implement
            abstract method")

    def stop(self):
        raise NotImplementedError("Subclass must implement
            abstract method")

class Sportscar(Car):
    def drive(self):
        return 'Sportscar driving!'

    def stop(self):
        return 'Sportscar braking!'
```

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

```
class Truck(Car):
    def drive(self):
        return 'Truck driving slowly because heavily loaded.
            '

    def stop(self):
        return 'Truck braking!'


cars = [Truck('Coal Truck'),
        Truck('Petroleum Truck'),
        Sportscar('Ferrari')]

for car in cars:
        print(car.name + ': ' + car.drive())
```

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Object Oriented Principles - Encapsulation

- Encapsulation is all about wrapping variables and methods in one single unit.
- Encapsulation is also known as data hiding. Why?
- Because, when you design your class you may (and you should) make your variables hidden from other classes and provide methods to manipulate the data instead.
- Your class is designed as a black-box.
- You have access to several methods from outside (classes) and a return type for each of those methods.
- Objects can hold crucial data for your application and you do not want that data to be changeable from anywhere in the code.

```
class Car:

    __maxspeed = 0
    __name = ""

    def __init__(self):
        self.__maxspeed = 200
        self.__name = "Supercar"

    def drive(self):
        print 'driving. maxspeed ' + str(self.__maxspeed)

    def setMaxSpeed(self,speed):
        self.__maxspeed = speed

redcar = Car()
redcar.drive()
#redcar.__maxspeed = 10
redcar.setMaxSpeed(320)
redcar.drive()
```

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

References

- https://javatutorial.net
- https://pythonspot.com
- Image source: Objects and Classes
- https://www.geeksforgeeks.org/classes-objects-java/