Lecture 1

What is Object Oriented Programming?

- Older programming languages like COBOL and C followed the Procedural Programming approach. The program written using these languages used to be a series of step by step instructions. They made use of procedures/subroutines for making the program modular. This programming paradigm focused on logic more than data.
- Modern programming languages like Java, C# etc. follow the Object Oriented approach. In object oriented programming, importance is given to data rather than just writing instructions to complete a task. An object is a thing or idea that you want to model in your program. An object can be anything, example, employee, bank account, car etc.

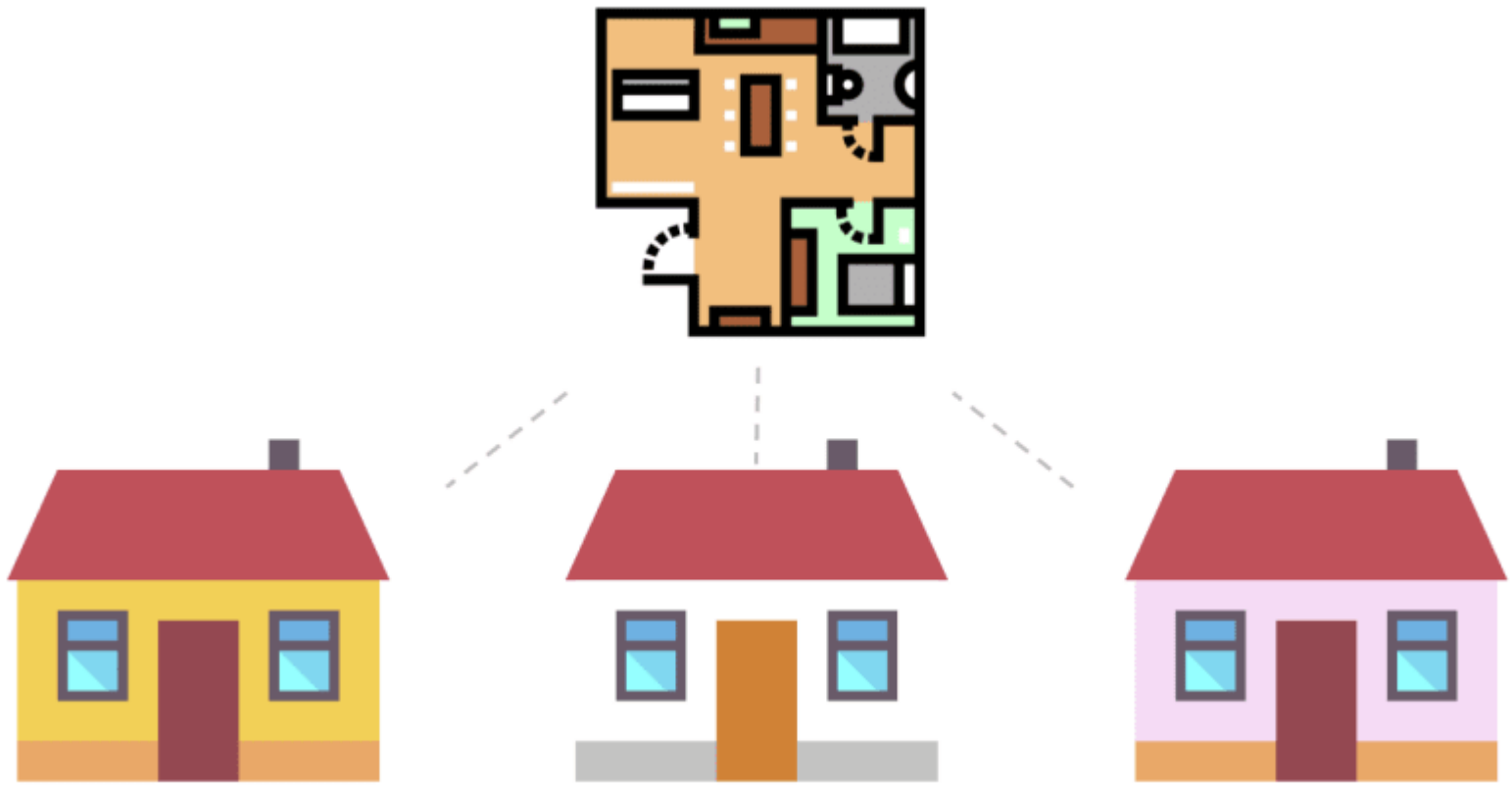
Advantages of OOP over structured programming

- It is easy to model a real system as real objects are represented by programming objects in OOP. The objects are processed by their member data and functions. It is easy to analyze the user requirements.
- With the help of inheritance, we can reuse the existing class to derive a new class such that the redundant code is eliminated and the use of existing class is extended. This saves time and cost of program.
- In OOP, data can be made private to a class such that only member functions of the class can access the data. This principle of data hiding helps the programmer to build a secure program that can not be invaded by code in other part of the program.
- With the help of polymorphism, the same function or same operator can be used for different purposes. This helps to manage software complexity easily.
- Large problems can be reduced to smaller and more manageable problems. It is easy to partition the work in a project based on objects.

Class, Object.. what's that?

For getting started with object oriented programming we would have to know what is a class and object and the difference between them. A class is a blueprint for creating an object. It is similar to the blue print of a house.

(Fig. is on the next page.)



An Example

A class defines *attributes and behavior*. If we are to model a car in our application then we could define attributes of the car like, model, fuel, make and behaviors like start, break, accelerate etc

```
public class Car{
    private string _color;
    private string _model;
    private string _makeYear;
    private string _fuelType;

    public void Start(){
        ..
    }

    public void Stop(){
        ..
    }

    public void Accelerate(){
        ..
    }
}
```

Using the same class "Car" we can create different objects having variation in model, fuel type and make year while having the same common behavior.

Object 1	
Model	Volkswagen Polo
Fuel	Petrol
Make	2017
Start() Break() Accelerate()	

Object 2	
Model	Volkswagen Vento
Fuel	Diesel
Make	2017
Start() Break() Accelerate()	

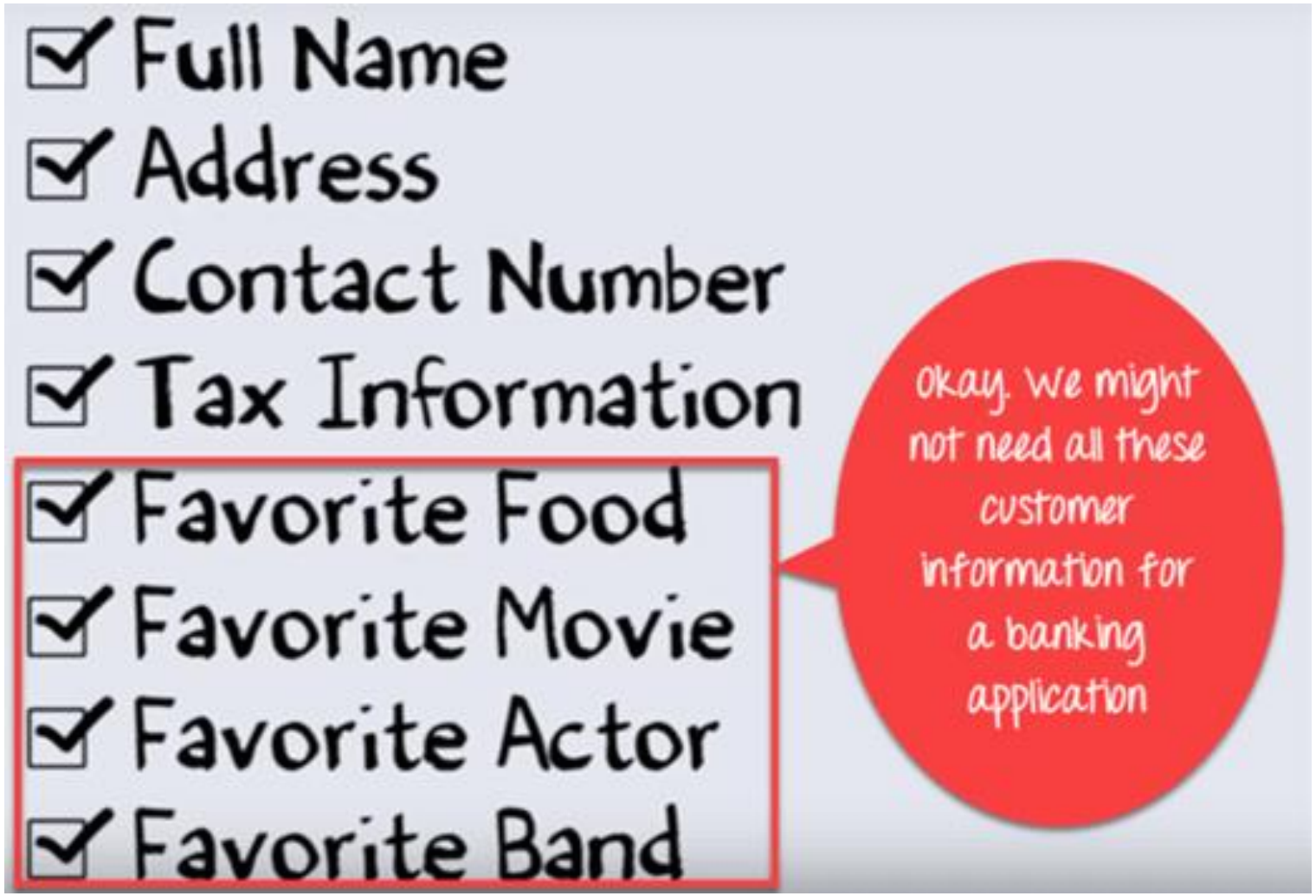
Abstraction

- Abstraction lets you focus on what the object does instead of how it is done.
- Abstraction means to represent the essential feature without detailing the background implementation or internal working detail.

=> Suppose you want to create a banking application and you are asked to collect all the information about your customer. There are chances that you will come up with following information about the customer(Figure in the next page)

But, not all of the above information is required to create a banking application.

So, you need to select only the useful information for your banking application from that pool. Data like name, address, tax information, etc. make sense for a banking application


Full Name
Address
Contact Number
Tax Information
Favorite Food
Favorite Movie
Favorite Actor
Favorite Band
okay. we might not need all these customer information for a banking application

Inheritance

Inheritance is a powerful feature of Object oriented programming languages. Inheritance helps in organizing classes into a hierarchy and enabling these classes to inherit attributes and behavior from classes above in the hierarchy.

Inheritance describes an “IS A relationship. This is how we talk in the real world. Example. A Parrot is a bird. US Dollar is a type of currency. But the phrase, Bank is a bank account is not correct.

Definition: Inheritance is a mechanism for code reuse and can help in reducing duplication of code.

Parrot

“IS A”
Bird

USD

“IS A”
Currency

Bank

“IS A”
Bank Account

Polymorphism

Polymorphism is the concept that there can be many different implementations of an executable unit and the difference happen all behind the scene without the caller awareness.

Method Overriding

Inheritance is one means of achieving polymorphism where behavior defined in the inherited class can be overridden by writing a custom implementation of the method. This is called method overriding also known as Run Time polymorphism.

```
public class WritingInstrument{

    public void Write(){
        // Makes visible mark
    }
}
```

```
public class Pen : WritingInstrument{

    public void Write(){
        // Makes visible mark with ink
    }
}
```

Method Overloading

There is also one more form of polymorphism called method overloading where inheritance does not come into the picture. The method name is same but the arguments in the method differ.

```
public class WritingInstrument{

    public void Write(){
        // Makes visible mark
    }

    public void Write(int fontSize){
        // Makes visible mark of given font
        // size
    }
```

Encapsulation

Encapsulation is:

- Binding the data with the code that manipulates it.
- It keeps the data and the code safe from external interference

Example with explanation-

Power steering of a car is a complex system, which internally have lots of components tightly coupled together, they work synchronously to turn the car in the desired direction. It even controls the power delivered by the engine to the steering wheel. But to the external world there is only one interface is available and rest of the complexity is hidden. Moreover, the steering unit in itself is complete and independent. It does not affect the functioning of any other mechanism.