OO Programming Languages

Elena Punskaya, elena.punskaya@eng.cam.ac.uk

OO in Java

- **Abstract** class defines shared implementation
- Methods declared Abstract **MUST** be implemented by subclasses
- **Interface** can only define method signatures (names, parameters, output) and no implementation
- Classes can *Extend* other classes and *Implement* interfaces
- By default, all class methods can be overridden (extended/implemented) in the subclasses

```
//abstract class declaration
 public abstract class Account
 {
      private int mAccountNumber;
      //constructor
      public Account(int accountNumber)
      {
           mAccountNumber = accountNumber;
      }
      // abstract method declaration
      public abstract void credit(Amount amount);
 }

 //interface declaration
 public interface IVerifiable
 {
      //declaring the interface method
      public boolean isVerified();
 }

 //PayPal derives from Account and realises IVerifiable
 public class PayPalAccount extends Account implements IVerifiable
 {
      //constructor, calls the superclass
      public PayPalAccount(int accountNumber)
      {
           super(accountNumber);
      }
      //implementation of the abstract method
      public void credit(Amount amount)
      {
           //send money to PayPal
      }
      //implementation of the interface method
      public boolean isVerified()
      {
           //do check and return result
      }
 }
```

OO in C#

- Very similar to Java syntax and concepts, but also some differences
- Any non-abstract class methods have to be explicitly declared virtual so can be overridden (extended/ implemented) in the subclasses
- C# vs Java comparison:
  - http://msdn.microsoft.com/en-us/library/ms836794.aspx

```
//abstract class declaration
 public abstract class Account
 {
      private int mAccountNumber;
      //constructor
      public Account(int accountNumber)
      {
           mAccountNumber = accountNumber;
      }
      // abstract method declaration
      public abstract void credit(Amount amount);
 }

 //interface declaration
 public interface IVerifiable
 {
      //declaring the interface method
      public boolean isVerified();
 }

 //PayPal derives from Account and realises IVerifiable
 public class PayPalAccount extends :Account, implements IVerifiable
 {
      //constructor, calls the superclass
      public PayPalAccount(int accountNumber) : base(accountNumber)
      {
          super(accountNumber);
      }
      //implementation of the abstract method
      public void credit(Amount amount)
      {
           //send money to PayPal
      }
      //implementation of the interface method
      public boolean isVerified()
      {
           //do check and return result
      }
 }
```

OO in C++

- Methods need to be declared virtual to be extended (as in C#)
- Pure virtual methods (ending declarations with “=0”) are equivalent to abstract methods in Java
- No dedicated concept of Interfaces, but same effect is achieved by defining a class that contains pure virtual methods only
- C++ and Java differences
  - http://www.cprogramming.com/tutorial/java/syntax-differences-java-c++.html

```
//class declaration
 class Account
{
      //declaring all publicly accessible methods/attributes
      public:
      //constructor
      Account(int accountNumber)
      {
           mAccountNumber = accountNumber;
      }
      // abstract method declaration
      virtual void credit(Amount amount) = 0;

      private:
          int mAccountNumber;
};

 //interface (equivalent) declaration
 class IVerifiable
{
      //pure virtual (abstract) method declaration
      public:
  virtual bool isVerified() = 0;
}

 //PayPal derives from Account and IVerifiable
public class PayPalAccount : public Account, public IVerifiable
{
      public:
      //constructor, calls the superclass
      PayPalAccount(int accountNumber):Account(accountNumber)
      {
      }
      //declaring the implementation
      virtual void credit(Amount amount);

      //declaring the implementation
      virtual bool isVerified();
}
```

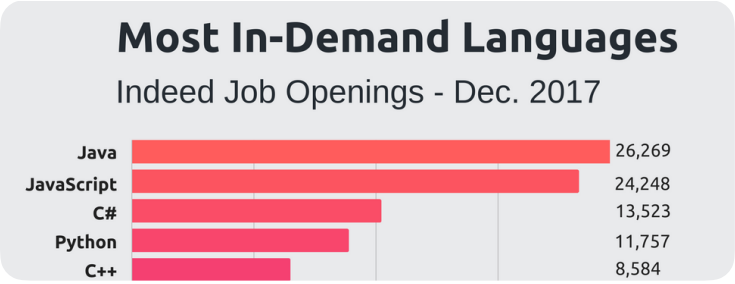
Summary

- Most modern languages incorporate some OO concepts
- Generally, those concepts are consistent across languages, however always worth checking
  - it may look like a Duck, walk like a Duck but doesn't quack like one :)
- Using OO features is the tool to get the job done, not the goal in itself
  - it's equally possible to write to write non-OO code in a very OO feature rich language and to write OO code in a less OO supportive one

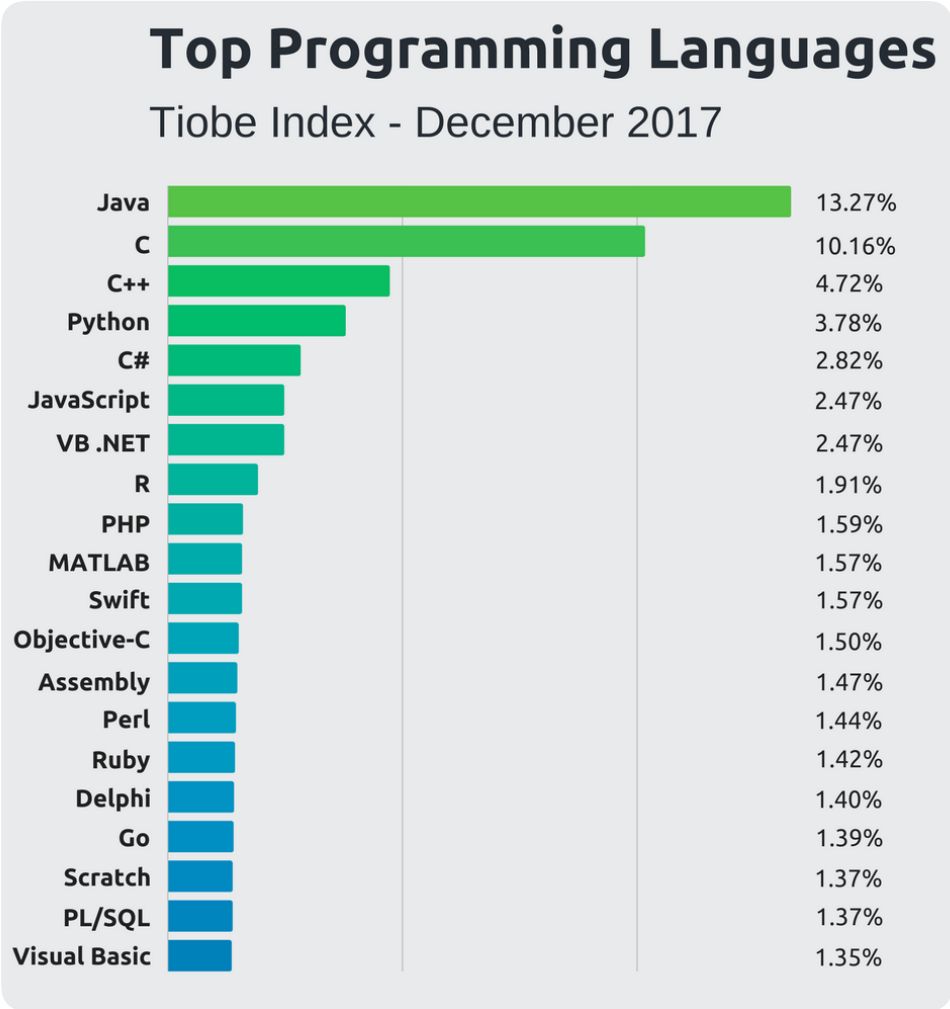
- And one more thing…

No Code Wars!

- Debating comparative advantages and disadvantages of programming languages makes a good (if often heated) conversation, but in reality the choice of the language is often dictated by the application!
- For example, in mobile application development:
  - Android: Java
  - iPhone: Swift/Objective-C
  - Windows Phone (RIP): C#
  - Symbian (RIP): C++
- Being proficient in a range of languages helps



Source: https://stackify.com/popular-programming-languages-2018/



Source: https://stackify.com/popular-programming-languages-2018/