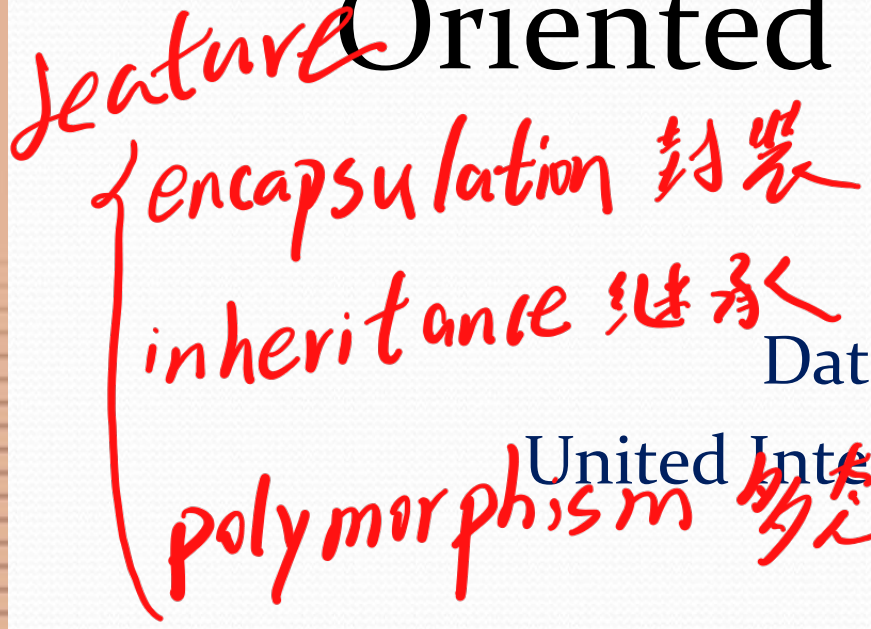
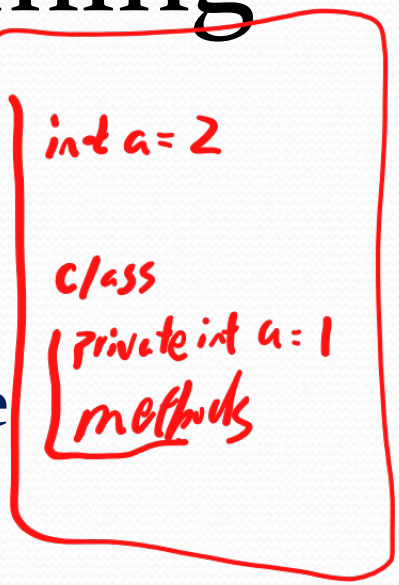
# Object-Oriented Programming


# Introduction to Object-Oriented Programming




Data Science

United International College



# Outline

- Introduction to Programming
- Object-Oriented Programming
- Objects vs. Classes

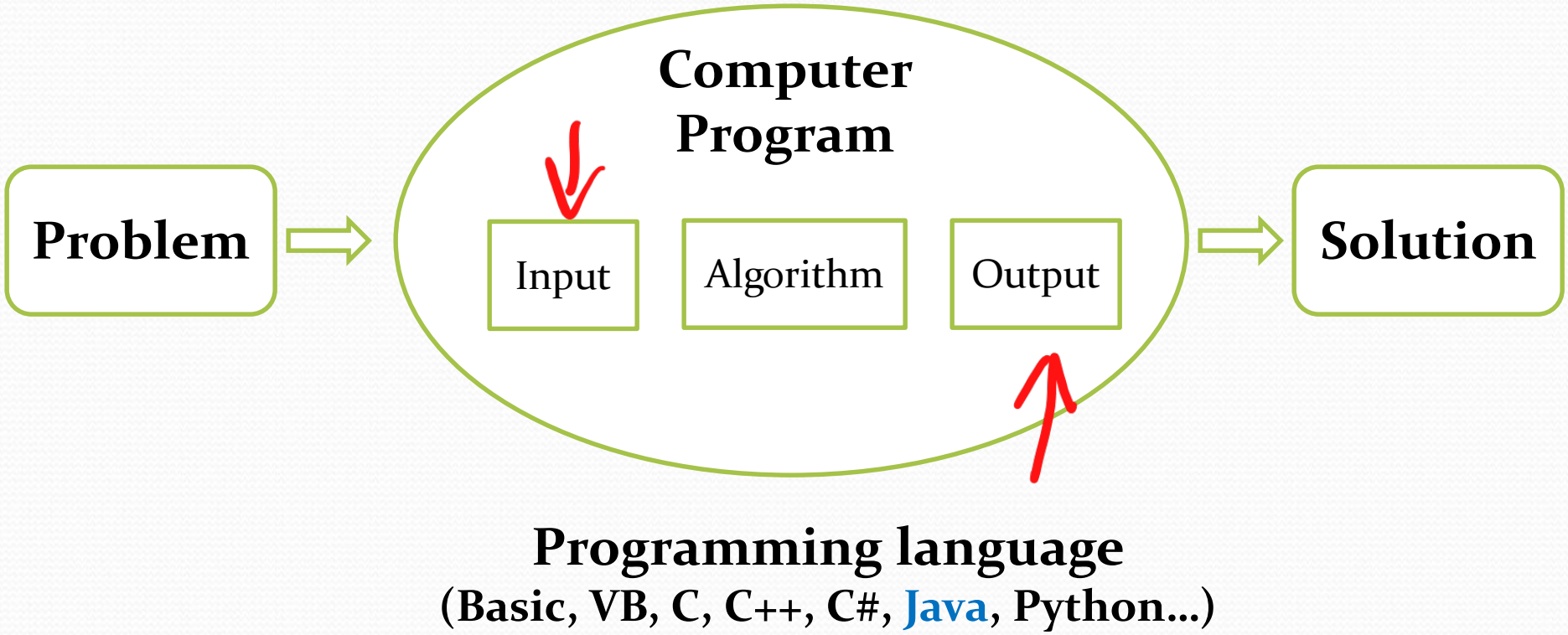
# Computer Program

- A computer is a tool for solving problems with data
- A computer program is a sequence of instructions that tell a computer how to do a task. When a computer follows the instructions in a program, we say it executes the program
  - Very similar to recipe
- Every computer program is an algorithm

# Programming Languages

- A programming language is a language, developed to express programs that implement specific algorithms
- Programming languages consists of a set of instructions for computers

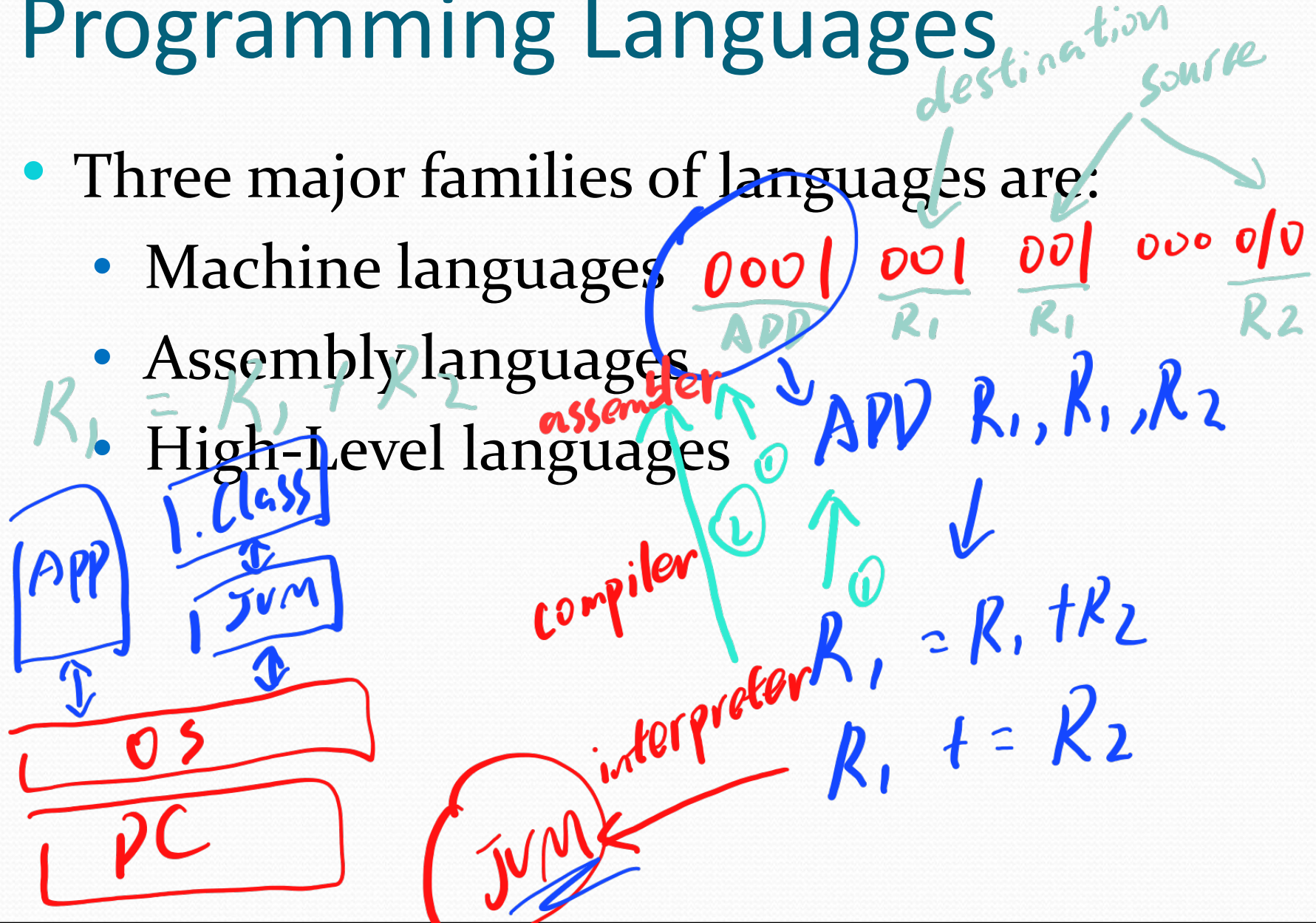
# Programming Languages




**Programming language**
**(Basic, VB, C, C++, C#, Java, Python…)**

# Programming Languages

- Three major families of languages are:
  - Machine languages
  - Assembly languages
  - High-Level languages



# Machine Languages

- Comprised of 1s and 0s (machine codes)
- "Native" language of a computer which they can understand
- Difficult to follow and program
- Example of code:
  - 1110100010101
  - 10111010110100

# Assembly Languages

- Assembly languages are a step towards easier programming
- Comprised of a set of elemental commands tied to a specific processor
- Needs to be translated to machine language before the computer processes it
- Example of code:
  - ADD ax, bx [Intel x86 style]

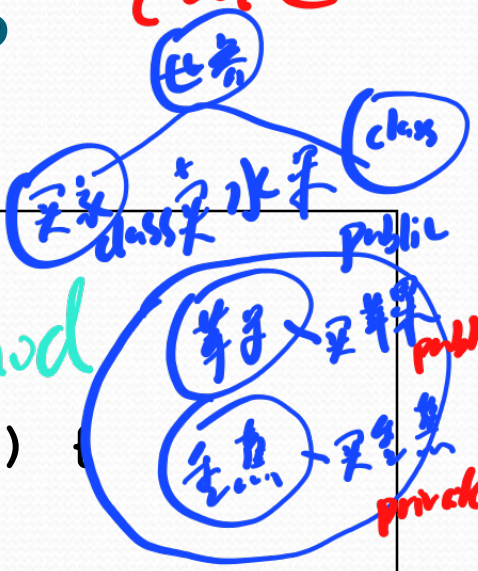
# High-Level Languages

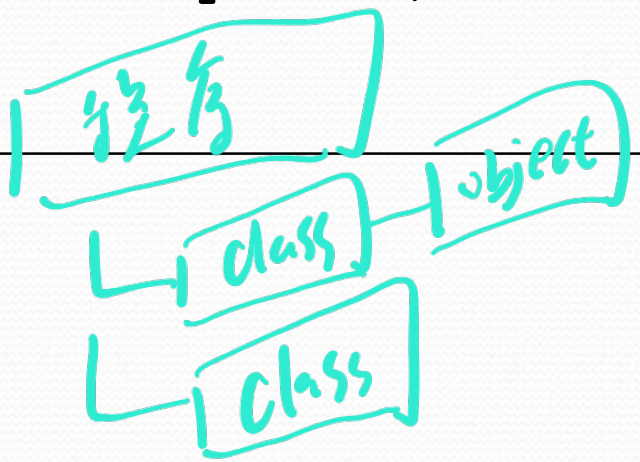
- A giant leap towards easier programming
- In contrast to assembly programming languages, it may use natural language elements, which will be easier to use
- Example of code:
  - grossPay = basePay + overTimePay

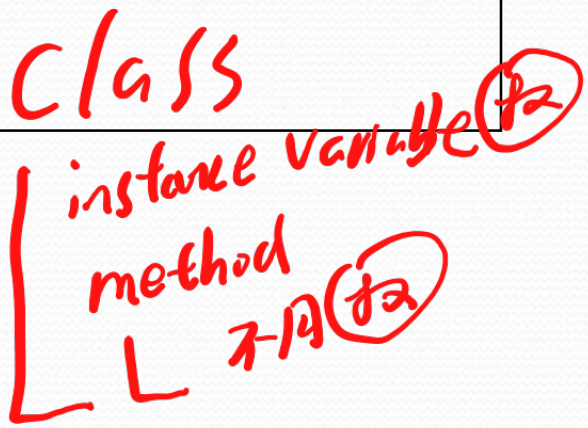
# Programming Languages

```
public class GrossPay
{

  public static void main (String[] args) {
        int basePay = 300;
        int overTimePay = 150;
        int grossPay = basePay + overTimePay ;
        System.out.printf("GrossPay:%d%n",grossPay);
   }
}
```



# Programming Languages

Class keyword

```
public class GrossPay
{

  public static void main (String[] args) {
       int basePay = 300;
       int overTimePay = 150;
       int grossPay = basePay + overTimePay ;
       System.out.printf("GrossPay:%d%n",grossPay);
  }
}
```

# Programming Languages

Class Name

```
public class GrossPay
{

  public static void main (String[] args) {
       int basePay = 300;
       int overTimePay = 150;
       int grossPay = basePay + overTimePay ;
       System.out.printf("GrossPay:%d%n",grossPay);
  }
}
```

# Programming Languages

Access Modifier, control the level of access other parts of a program have to this code

```
public class GrossPay
{
  public static void main (String[] args) {
       int basePay = 300;
       int overTimePay = 150;
       int grossPay = basePay + overTimePay ;
       System.out.printf("GrossPay:%d%n",grossPay);
  }
}
```

# Programming Languages

```
public class GrossPay
{
  public static void main (String[] args) {
       int basePay = 300;
       int overTimePay = 150;
       int grossPay = basePay + overTimePay ;
       System.out.printf("GrossPay:%d%n",grossPay);
  }
}
```

Curly braces pair enclose a block of code. Class level scope

# Programming Languages

This is a method of class GrossPay, named Main

```
public class GrossPay
{
  public static void main (String[] args) {
       int basePay = 300;
       int overTimePay = 150;
       int grossPay = basePay + overTimePay ;
       System.out.printf("GrossPay:%d%n",grossPay);
  }
}
```

# Programming Languages

Variables, which are used to store data

```
public class GrossPay
{
  public static void main (String[] args) {
       int basePay = 300;
       int overTimePay = 150;
       int grossPay = basePay + overTimePay ;
       System.out.printf("GrossPay:%d%n",grossPay);
  }
}
```

args argc argv

# Programming Languages

Data types

```
public class GrossPay
{
  public static void main (String[] args) {
       int basePay = 300;
       int overTimePay = 150;
       int grossPay = basePay + overTimePay ;
       System.out.printf("GrossPay:%d%n",grossPay);
  }
}
```

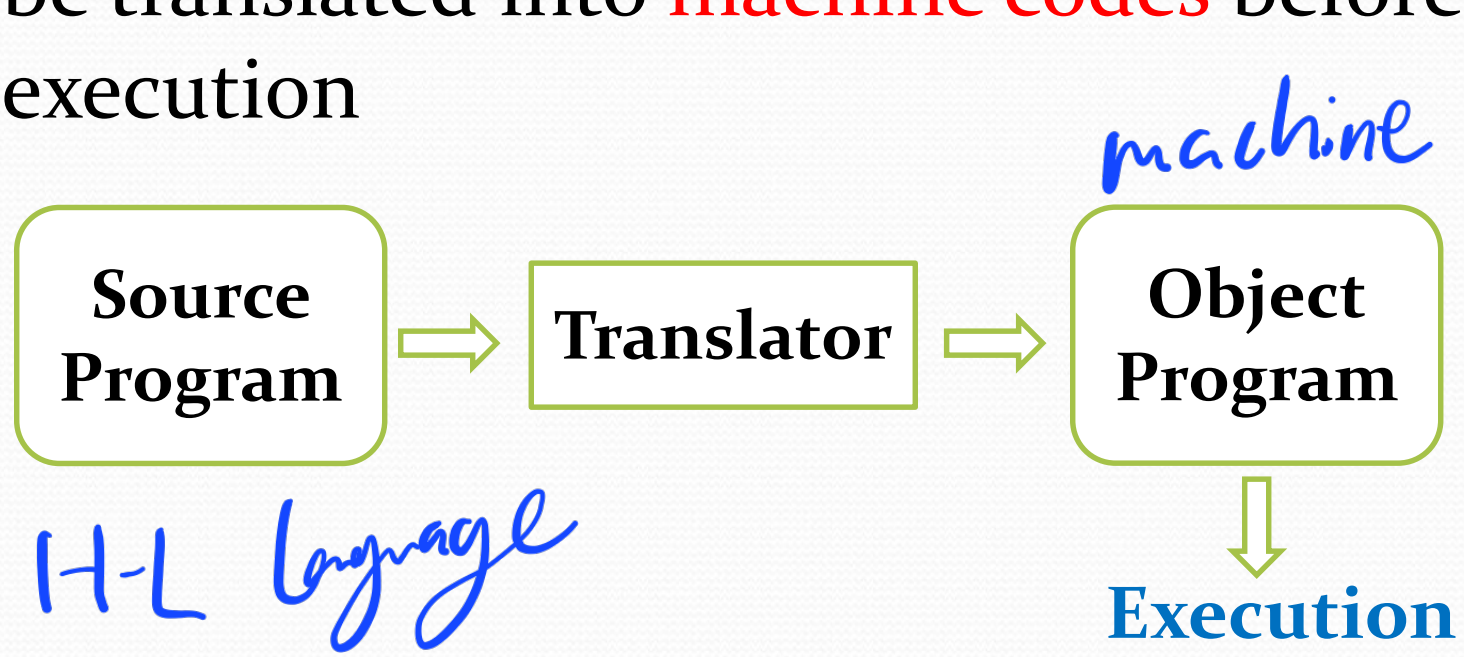
# Programming Languages

```
public class GrossPay
{
  public static void main (String[] args) {
       int basePay = 300;
       int overTimePay = 150;
       int grossPay = basePay + overTimePay ;
       System.out.printf("GrossPay:%d%n",grossPay);
  }
}
```

Statement, forms a complete command to be executed.

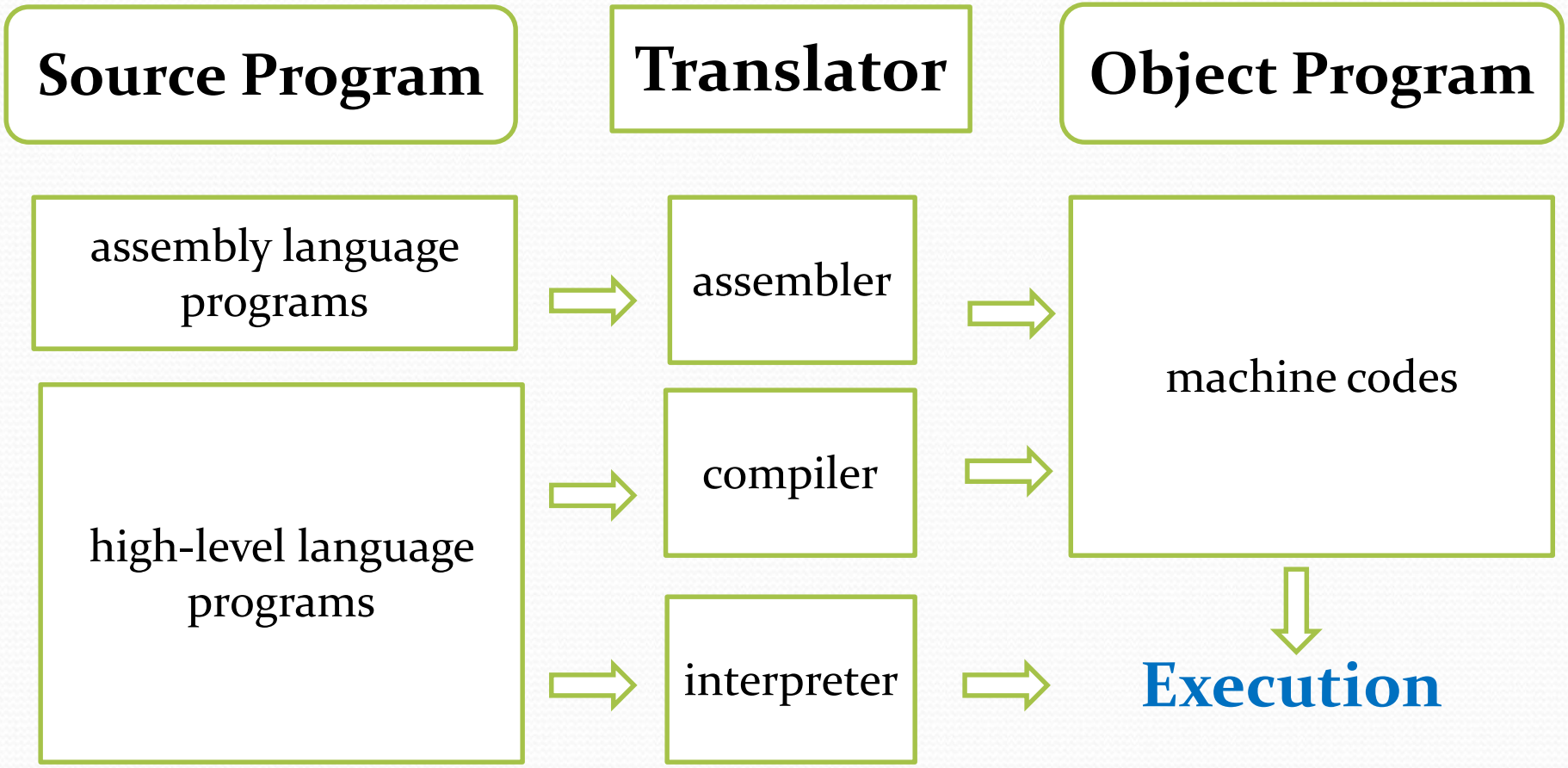
# Translators

- Programs (except machine language) must be translated into machine codes before execution




*A simplified translation process*

# Translators




*The functions of three types of translators*

# High-Level Languages

- Historically, the high-level languages can be divided into two groups: procedural languages and object-oriented languages
- Procedural language
  - specifies a series of well-structured steps and procedures within its programming context to compose a program
  - Examples include C, Fortran, Perl, HTML, …

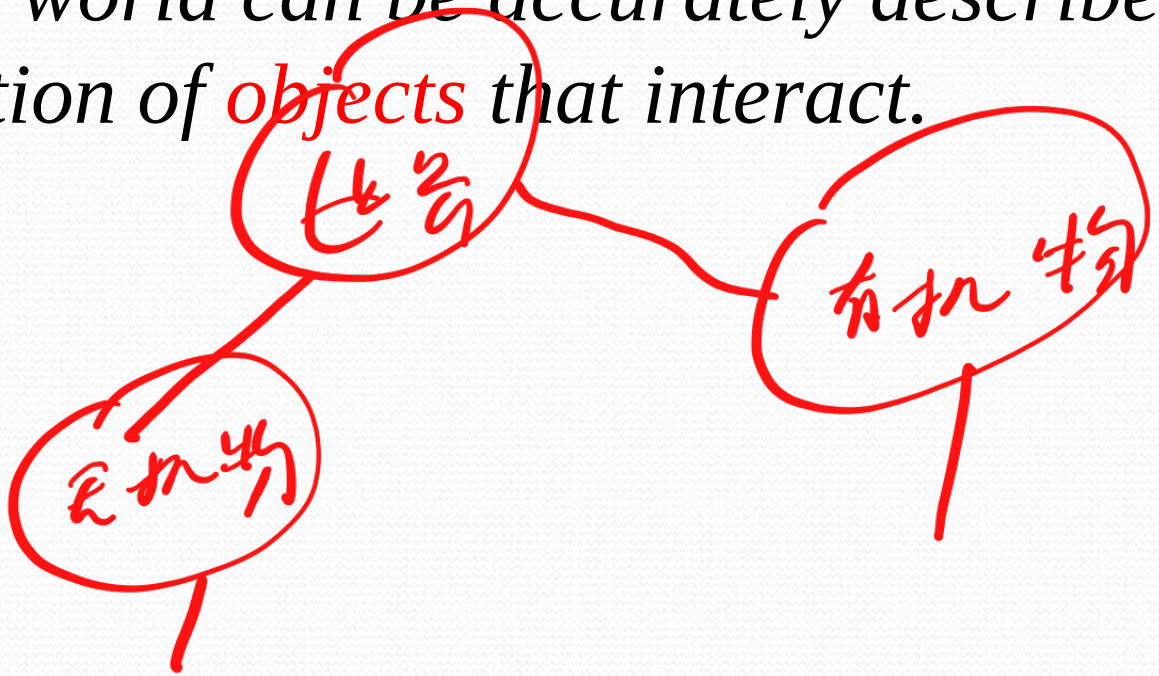
# Object-Oriented Programming

- Object-oriented programming is the dominant language paradigm these days.
- The focus of OOP is on modeling data
- An object-oriented program is made of objects, the class is the blueprint from which the objects are made.
- Examples of OOP language include C++, Visual Basic.Net, Java, Python.

# Object-Oriented Programming

Key idea in OOP:

*The real world can be accurately described as a collection of objects that interact.*

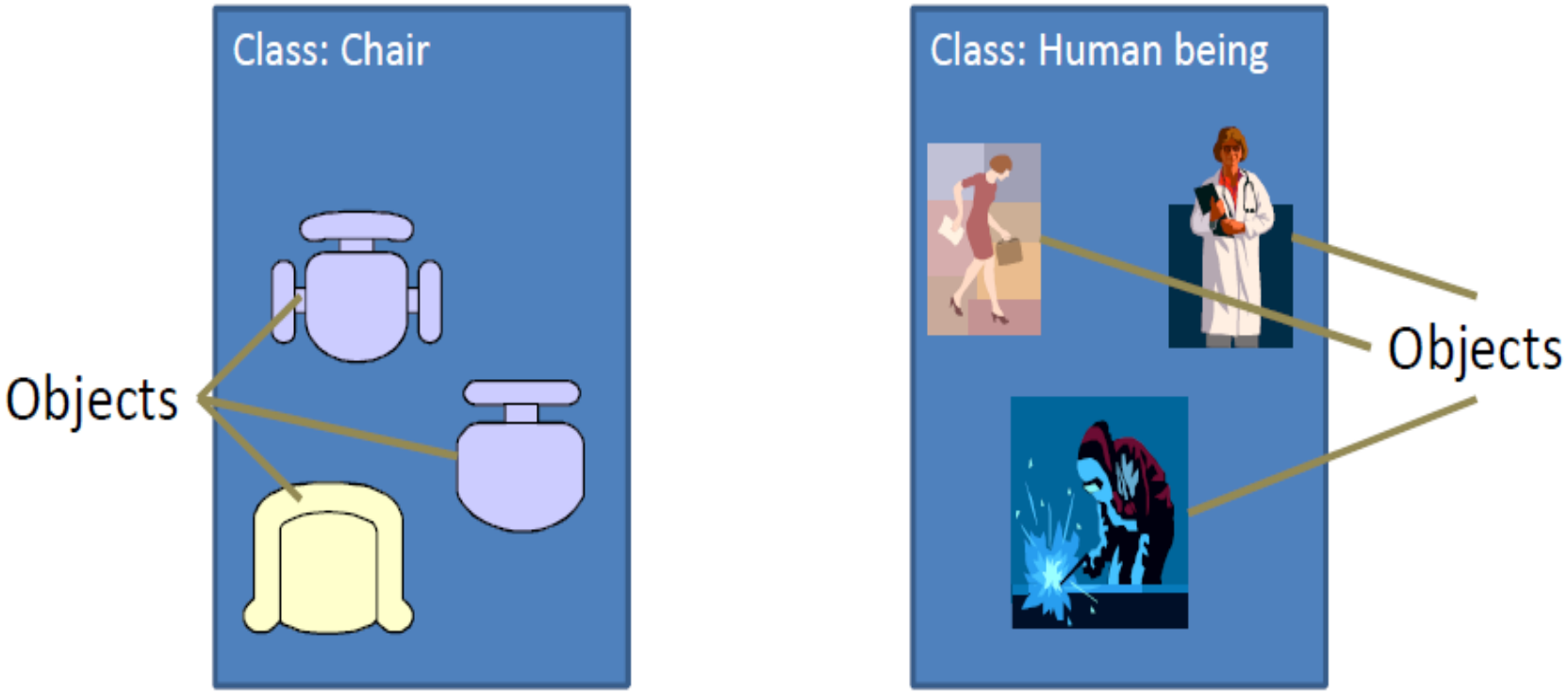


# Classes and Objects

- A class is  a prototype, template and blueprint from which objects are made.
- An object is an instance（实例） of a class.
- A class is a generalized description that describes a collection of similar objects.

- Example: Chair, Human, Students, Teachers, Books, etc.

# Classes and Objects



# Classes and Objects

- Object - An entity
  - Physical: a chair, a desk, a person
  - Logical: a list, a stack, a rectangle
- Objects have **data** and **behavior**
  - Data in an object is called **instance fields**
  - Behaviors that operate on the data are called **methods**
  - A specific object will have a specific values for its instance fields. The set of those values is the current **state** of the object.
- Example: Dog
  - **Data**: Color, Name, Breed
  - **Behaviors**: Fetch stick, Drink water, Wag tail, Bark

# Classes and Objects

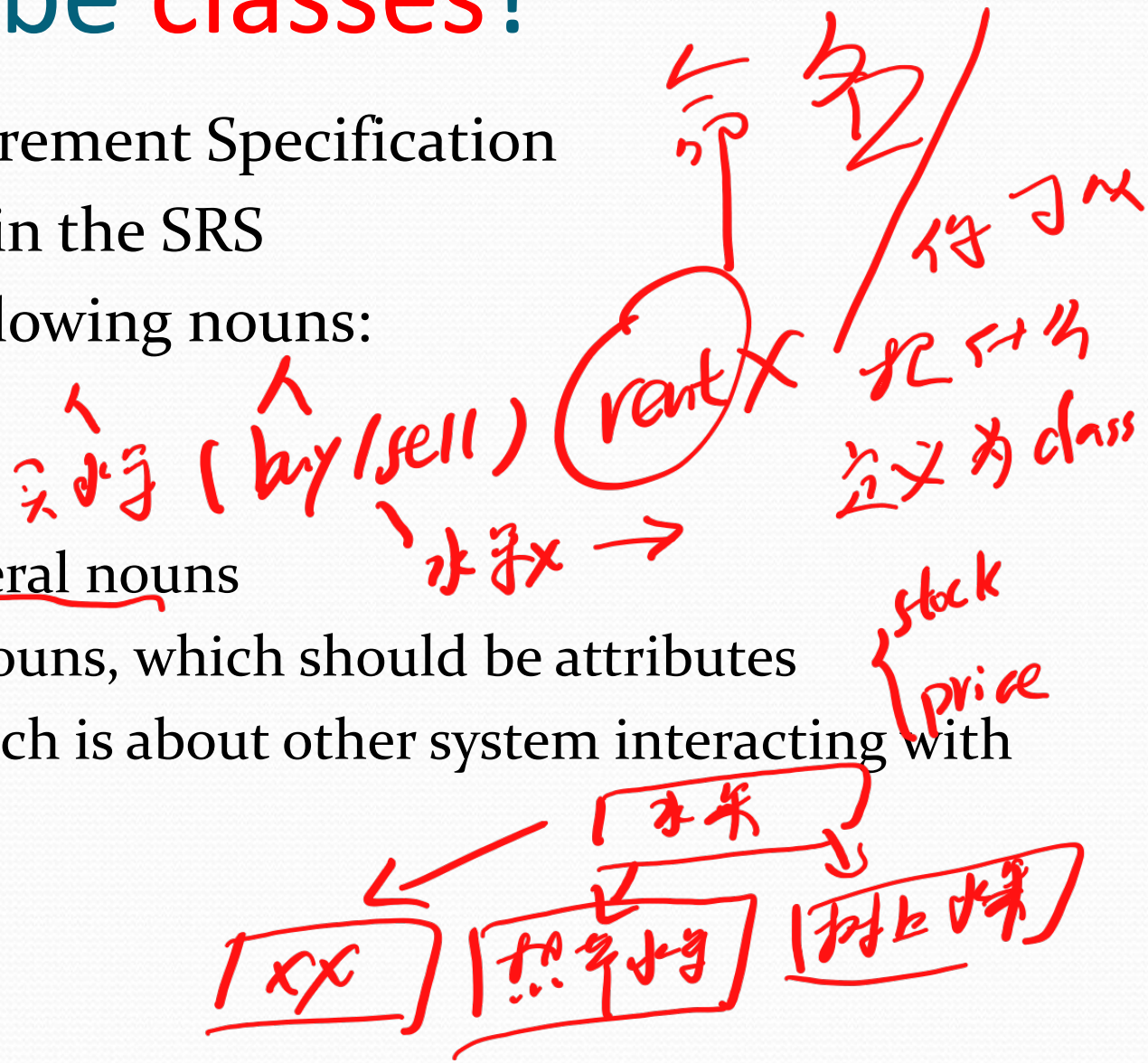
*An object is an instance of a class*

*A class is a blueprint from which objects are made*

- Animal is a class, a cat called "Snowball" is an object
- Vehicle is a class, my BMW is an object
- Student is a class, a student called "Tom" is an object
- …

# What can be classes?

- Software Requirement Specification
- Find all nouns in the SRS
- Remove the following nouns:
  - Duplicates
  - Unrelated
  - Vague or general nouns
  - Dependant nouns, which should be attributes
  - Interface, which is about other system interacting with the system



# Object-Oriented

- OO in one sentence:

  keep it **DRY**, keep it Shy and Tell the other guy

  - DRY: Do not Repeat Yourself concise
  - Shy: Should not reveal the information about itself unless really necessary encapsulation
  - Tell the other guy: Send a message rather than doing a function call. #1 API

- By Andy Hunt and Dave Thomas.

# Benefits of OOP Approach

- **Modularity**:
  - The source code for an object can be written and maintained independently of the source code for other objects.
- **Information-hiding**:
  - By interacting only with an object's methods, the details of its internal implementation are hidden from the outside world.
- **Code re-use**:
  - If an object already exists you can use that object in your program. This allows specialists to implement / test / debug complex, task-specific objects, which you can then trust to run in your own code.

From Java Tutorial -- http://java.sun.com/docs/books/tutorial/java/concepts/object.html

# Object-Oriented Software Development Process

- OO Analysis: Requirement specification
- OO Design: Architectural design
- Object Design: Detailed design
- **Object-Oriented Programming: Implementation**

Our focus this semester!

# Summary

- Introduction to Programming
- Object-Oriented Programming
- Objects vs. Classes