Chapter 6

Thinking Object-Oriented

Chapter 6 Topics

- **Language and Thought**
  - **Sapir-Whorf Hypothesis**
  - **Example from Computer Languages**
- **A New Paradigm(范例)**
  - **Definition of Paradigm**
  - **Kay's Description of Object-Oriented Programming**
  - **Illustration of OOP Concepts**

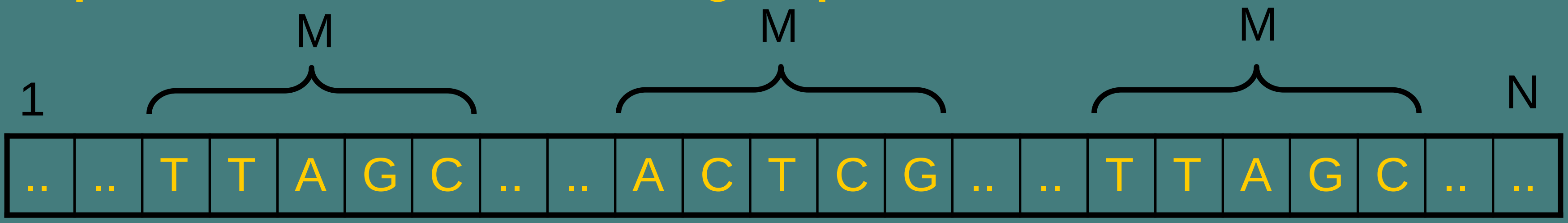
Sapir-Whorf Hypothesis

In linguistics（语言学） there is a hypothesis（假说） that the languages we speak directly influence the way in which we view the world.

- Eskimo (or Innuit) languages have different words to describe different types of snow
—wet, fluffy, heavy, icy, and so on.

Example from Computer Languages

A student working in DNA research had the task of finding repeated sequences of M values in a long sequence of N values:



..	..	T	T	A	G	C	..	..	A	C	T	C	G	..	..	T	T	A	G	C	..	..

Wrote the simplest FORTRAN program:

```
       DO 10 I = 1, N-M+1
       DO 10 J = 1, N-M+1
       FOUND = .TRUE.
       DO 20 K = 1, M
20  IF X[I+K-1] .NE. X[J+K-1] THEN FOUND = .FALSE.
       IF FOUND THEN ...
10  CONTINUE
```

A Better Solution

A friend writing in APL found a much better solution by rearranging the data and sorting（排序）.

x_1	x_2	x_3	…	x_{m-1}	x_m
x_2	x_3	x_4	…	x_m	x_{m+1}
x_3	x_4	x_5	…	x_{m+1}	x_{m+2}
…	…	…	…	…	…
x_{n-m+1}	x_{n-m+2}	x_{n-m+3}	…	x_{n-1}	x_n

What lead to the discovery?

Why did the APL programmer find the better solution?

- Fortran programmer was blinded by a culture that valued loops, simple programs
- Sorting is a built-in operation in APL, good programmers try to find novel uses for sorting

The fundamental point is that the language you speak leads you in one direction or another.

Chapter 6 Topics

- Language and Thought
  - Sapir-Whorf Hypothesis
  - Example from Computer Languages
- A New Paradigm
  - Definition of Paradigm
  - Kay's Description of Object-Oriented Programming
  - Illustration of OOP Concepts

Definition of Paradigm

Object-oriented programming is often described as a new *paradigm*.

Thomas Kuhn [1970] used the term to describe a set of theories, standards, and methods that represent a way of organizing knowledge—a way of viewing the world.

Kay's Description of Object-Oriented Programming

- Everything is an ***object*** （对象）.
- Objects perform computation by making requests of each other through the passing of ***messages***（*消息*）.
- Every object has it's own ***memory***, which consists of other objects.
- Every object is an ***instance***（*实例*） of a ***class***（*类*）. A class groups similar objects.
- The class is the **repository(仓库)** for ***behavior***（*行为*） associated with an object。
- Classes are organized into **singly-rooted**（单根） tree structure, called the ***inheritance hierarchy***（*继承层次*）.

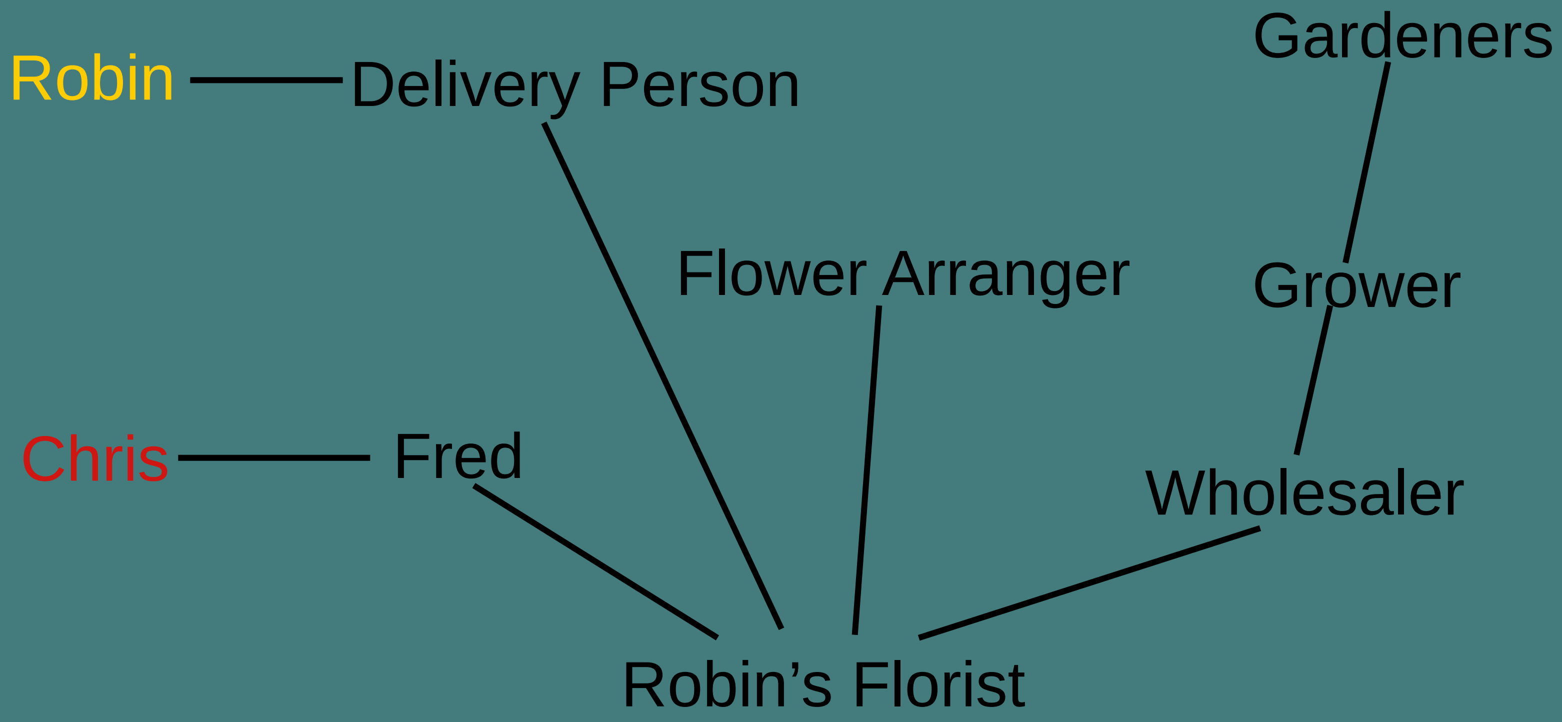
Illustration of OOP Concepts -- Sending Flowers to a Friend

Chris is sending flowers to Robin who lives in a different city.

Chris

Robin

Agents（代理） and Communities（团体）



Robin

Delivery Person

Gardeners

Flower Arranger

Grower

Chris

Fred

Wholesaler

Robin’s Florist

Elements of OOP - Objects

So we have Kay's first principle.

1. Everything is an *object*.

Actions in OOP are performed by agents, called *instances* or *objects*.

Examples:

Chris, Robin, the florist Fred, the florist in Robin’s city, the driver, the flower arranger, the grower,etc.

Elements of OOP - Messages

And principle number 2:

2. Objects perform computation by making requests of each other through the passing of *messages*

Actions in OOP are produced in response to requests for actions, called *messages*. An instance may accept a message, and in return will perform an action and return a value.

Examples:

To begin the process of sending the flowers, Chris gives a message to Fred. Fred in turn gives a message to the florist in Robin’s city, who gives another message to the driver, and so on.

Information Hiding

A user of a service being provided by an object

- need only know the name of the messages that the object will accept.
- need not have any idea how the actions, performed in response to the request, will be carried out.

Having accepted a message, an object is responsible for carrying it out.

Elements of OOP - Receivers

Messages differ from traditional function calls in two very important respects:

- In a message there is a designated ***receiver(接收者)*** that accepts the message
- The **interpretation（解释）** of the message may be different, depending upon the receiver

Elements of OOP– Recursive（递归的） Design

3. Every object has it's own *memory*, which consists of other objects.

Each object is like a miniature（微型的） computer itself - a specialized processor performing a specific task.

Elements of OOP - Classes

4. Every object is an *instance* of a *class*. A class groups similar objects.

5. The class is the repository for *behavior* associated with an object.

Behavior is associated with classes, not with individual instances. All objects that are instances of a class use the same method in response to similar messages.

Examples:

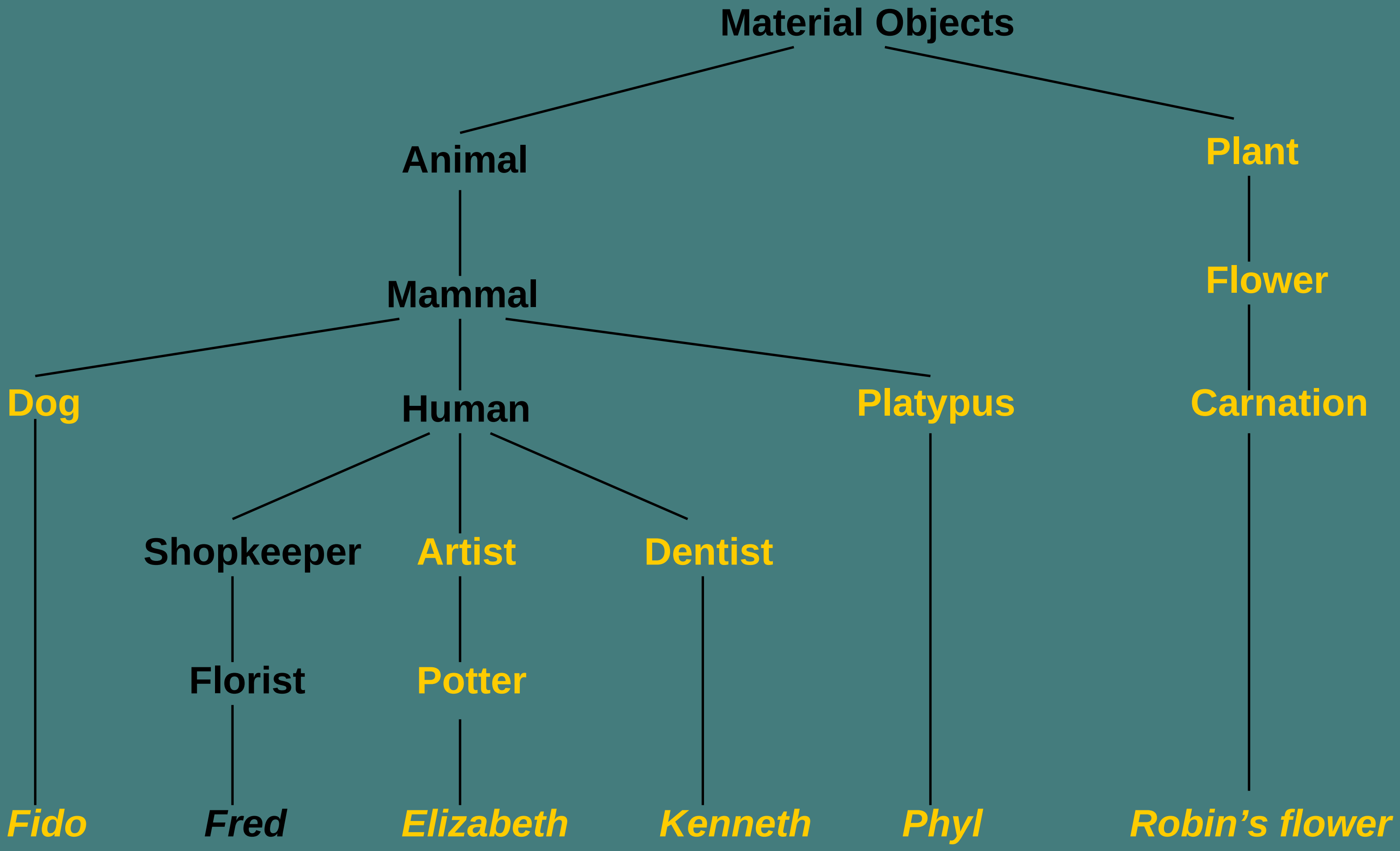
Fred is an *instance* of the *class* Florist

the behavior of Fred is determined from a general idea of Florists.

Hierarchies of Categories(种类)

- But Fred is not just a Florist. He is a Shopkeeper, a Human, a Mammal, a Material Objects, and so on.
- At each level of abstraction, certain information is recorded.

Class Hierarchies



Elements of OOP - Inheritance

6. Classes are organized into a singly-rooted tree structure, called an *inheritance hierarchy*

Information (data and/or behavior) associated with one level of abstraction in a class *hierarchy* is automatically applicable to lower (more specialized) levels.

Elements of OOP - Overriding

Subclasses can alter or ***override***（改写）information inherited from parent classes.

Examples:

- All mammals give birth to live young
- A **platypus(鸭嘴兽)** is an egg-laying mammal

Inheritance combined with overriding are where most of the power of OO originates.