程序设计实习（实验班-2023春）
面向对象编程：绪论

授课教师：姜少峰

助教：张宇博 楼家宁

Email: shaofeng.jiang@pku.edu.cn

我们为什么要关心各种编程范式?

- 本质上看，这些不会让能解决的问题范围变大
- 因为任何一个程序可以用任何其他的编程语言/方法/范式来实现

- 此处关注点在于：使得特定的编程任务变“简单”
- 这是一个主观的过程，就是要“人性化”

- 我们从大家熟知的C/Pascal等面向过程语言出发，来对比介绍/引出面向对象

面向过程?

- 主体是“过程”：
  - 我们一般说的解决某个问题的一段算法，就是一个“过程”
  - 整个程序是一个大过程，可以由若干小过程组成
  - 面向过程编程的哲学就是拆分成一个个子过程，各个击破
- “过程”是主体：面向过程的程序设计是由一个个细分的“过程”构成的
  - C语言是一个面向过程的语言（C++也支持这个子集）

面向过程的程序

- doA(X)
- doB(Y)
- doA(Z)
- doC(Y)
- …

面向对象的主体：“对象”

- 过程为主体：append(S, “abc”)将”abc”拼接在字符串S后面
- 对象为主体：S.append(“abc”)，变量S成为了主体，append函数是S“内部”的函数

主 谓 宾

- 对象 = 有自己状态和行为的实体
- 即：对象结合某些其他信息和自己的状态，实施了某种行为

类：“对象”的抽象

- 状态 = 数据，行为 = 函数
- 类 = 对象的抽象，对象 = 类的具体实例/变量

- C的结构体：没有函数，本质上仍是对“数据”抽象
- 类：结构体的数据之上，还含有一些依存于本类的函数
- 函数成为了类的一部分，类和类交互的主体就是类/对象，而不再是函数

对象为主体的程序

“主谓宾”

- A.doSth1(B)
- B.doSth2(C)
- B.doSth3(A)
- …

举例

- 刚刚提到的String的实现
- 字符串S是一个对象
- S对应的类应该包含数据和动作
  - 数据：所存储的字符串的内容，比如可以表示成一个char数组
  - 动作：append函数，以及潜在的其他函数比如search(String T)

好处?

- 建模的便利
  - 一般一个类都有对应的“物理”/“现实”语意，高度相关的数据机器操作放一起
  - 例如刚刚讲到的字符串，那么字符串操作都放在一个类里面比较自然
- 更重要的：封装
  - 实现“内外隔离”：外界尝试访问类的时候可以做到对类内细节的无知
  - 话句话说：当类需要与外界交互时，能自己维护好自己的状态

封装

例子：封装的重要性

- 考虑一个类A，里面维护一个取值在[1, 100]整数的数组arr
  - 支持：尾部插入，访问数组arr[i]，统计每个[1, 100]元素被访问多少次
- 假设有若干函数要从A取数据，也就是经常需要访问数组元素arr[i]
- 先考虑面向过程怎么实现？

例如最后统计heavy hitter

```
struct A
{
    int arr[10000]; int tail = 0; int freq[101];
}
```

```
void f1(A a)
{
    // ...
    for (i = ...)
    {
        res += a.arr[i];
        a.freq[a.arr[i]]++;
    }
    // ...
}
```

```
void f2(A a)
{
    // ...
    for (i = ...)
    {
        res = min(res, a.arr[i]);
        a.freq[a.arr[i]]++;
    }
    // ...
}
```

编程的额外负担；
逻辑上不必要的耦合、混乱

问题1：任何地方调用a.arr[i]的时候，都得特别注意维护好a.freq！

问题2：f1和f2甚至没有“义务”维护freq，因为他们的功能根本不需要用到freq！

封装：解决问题1

问题1：任何地方调用a.arr[i]的时候，都得特别注意维护好a.freq！

```
struct A
{
    int arr[10000]; int freq[101];

    int get(int i)
    {
        freq[arr[i]]++;
        return arr[i];
    }
};
```

外界需要arr[i]时需要调用get(i)，自动维护好内部的freq信息，外界不需要考虑维护freq！

- 除此以外：外界也不该直接访问arr和freq，这些也可以/应该封装
- 一般原则：类/对象要将自己的状态隐藏，与外界交互只通过动作

自己的状态自己维护好
和外界交互只通过做动作进行

状态 = 数据
动作 = 函数

封装：解决问题2

- 如果所有类都做了良好的封装，那么可以避免：

  问题2：f1和f2甚至没有“义务”维护freq，因为他们的功能根本不需要用到freq！

- 一般来说，耦合度过高的问题是：
  - 实现某功能时不能专注于实现当前功能，还需要考虑对其他功能的影响
- 封装：将“不必要”的细节隐藏，外界交互只需专注本身要实现的功能

更高的可维护性

- 回忆考虑一个类A，里面维护一个取值在[1, 100]整数的数组arr
  - 支持：尾部插入，访问数组arr[i]，统计每个[1, 100]元素被访问多少次
- 现在加了一个需求，还要维护arr[i]的min、max等统计量

```
struct A
{
    int arr[10000]; int tail = 0; int freq[101];
}
```

面向过程：这些地方全都要加上min、max的维护！

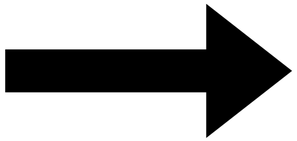
```
void f1(A a)
{
    // ...
    for (i = ...)
    {
        res += a.arr[i];
        a.freq[a.arr[i]]++;
    }
    // ...
}
```

```
void f2(A a)
{
    // ...
    for (i = ...)
    {
        res = min(res, a.arr[i]);
        a.freq[a.arr[i]]++;
    }
    // ...
}
```

```cpp
struct A
{
    int arr[10000]; int freq[101];

    int get(int i)
    {
        freq[arr[i]]++;
        return arr[i];
    }
};
```

只统一修改get函数即可



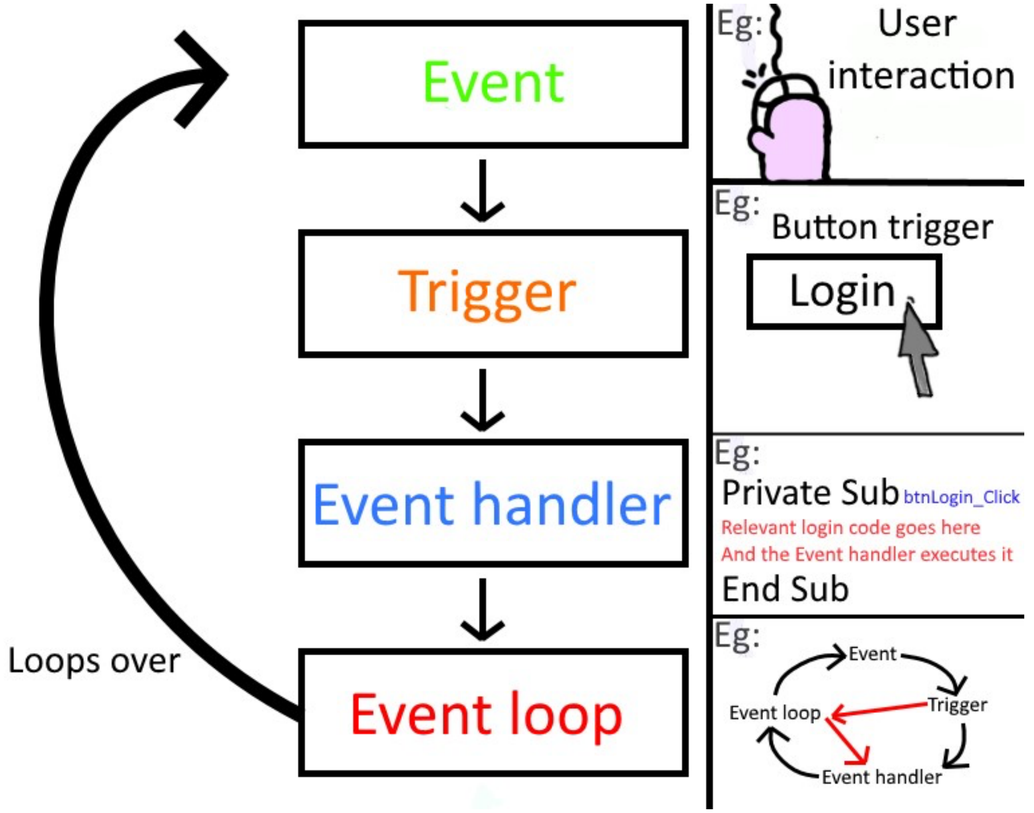
```cpp
struct A
{
    int arr[10000]; int freq[101];

        int get(int i)
    {
        freq[arr[i]]++;
        min = std::min(min, arr[i]);
        max = std::max(max, arr[i]);
        return arr[i];
    }
};
```

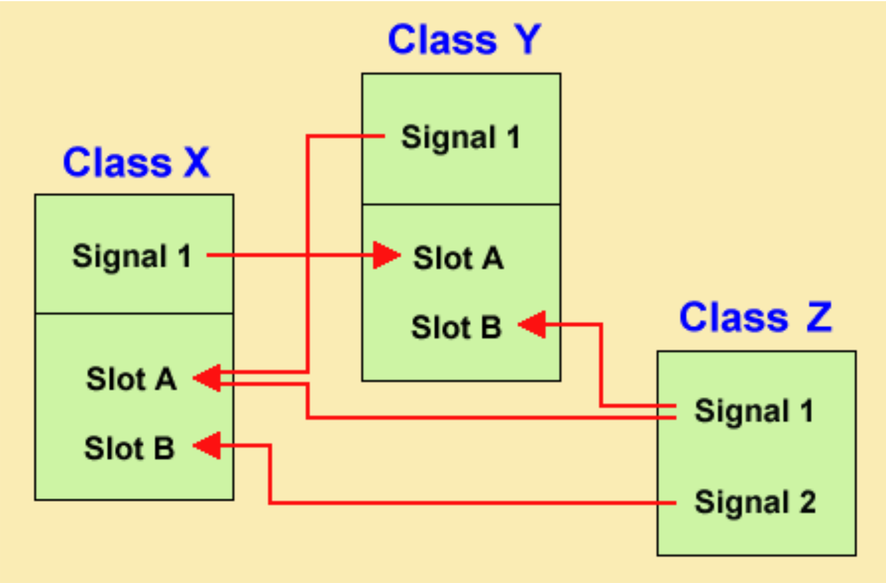
数据流 -> 信息流

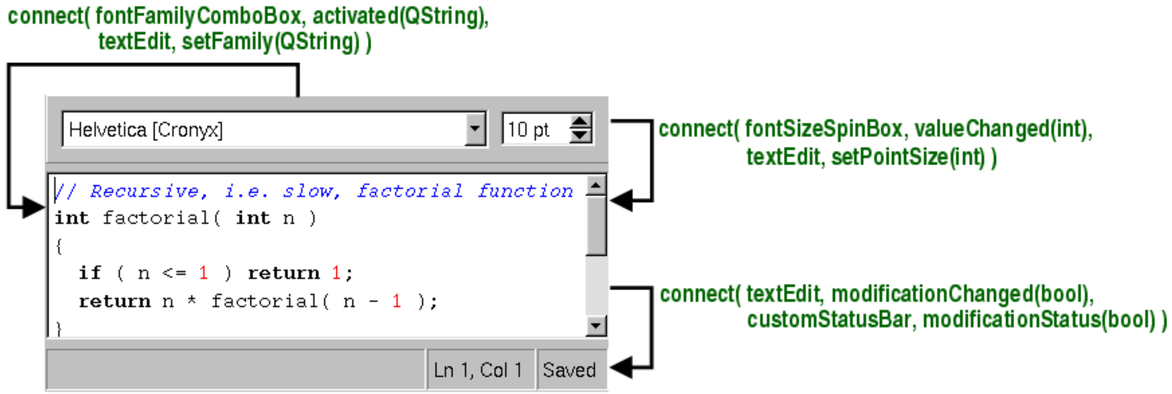
- 面向对象传递的是struct，也就是数据
- 面向对象隐藏了数据，整个程序就由动作的反复触发构成
- 动作被什么驱动?
  - 信息（在一些场景下叫做事件）
- 因此，面向对象编程将数据如何交互，变成了信息如何传递、触发动作

例子：基于事件的GUI编程



例子：Qt的signal-slot机制



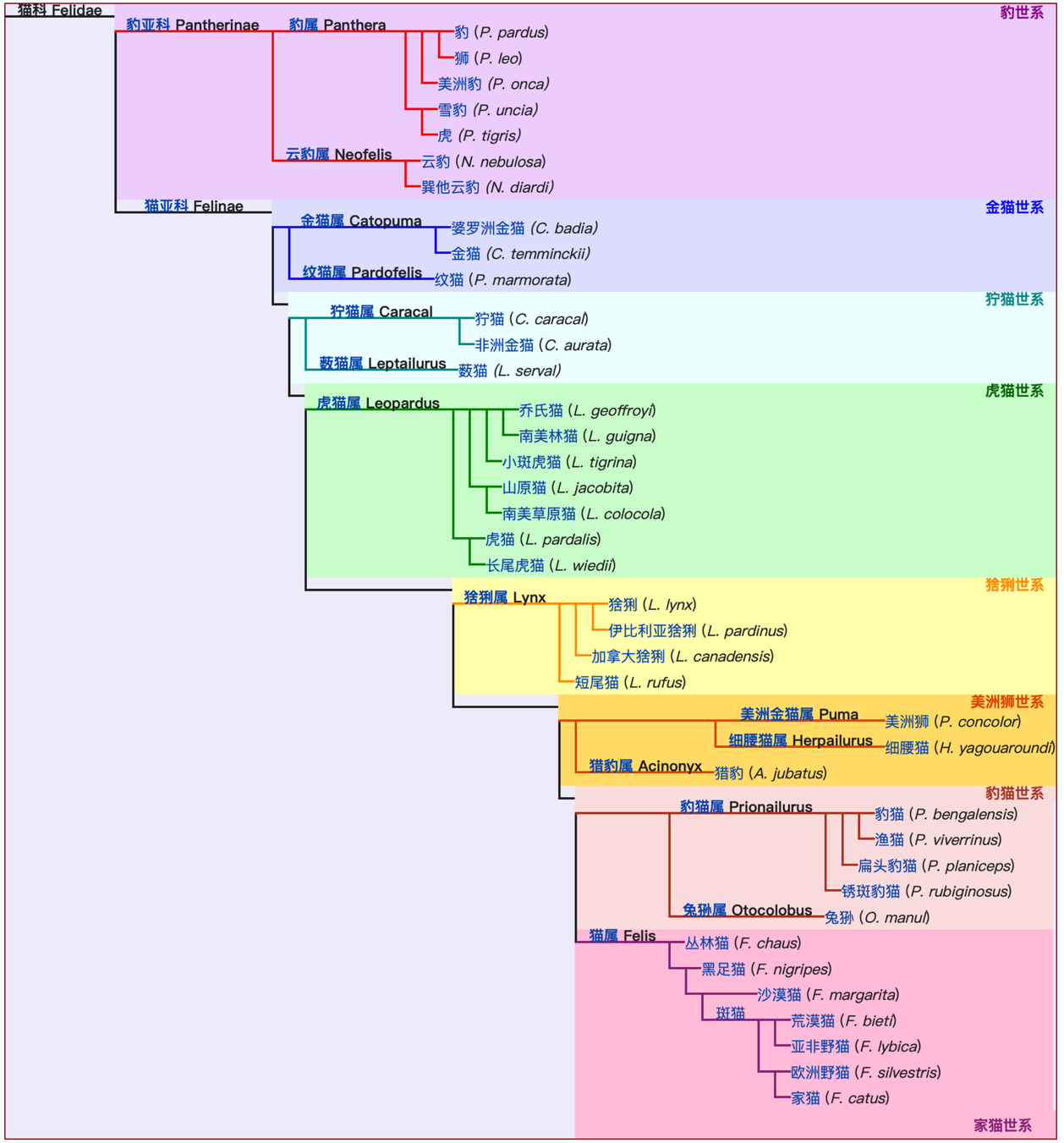


继承

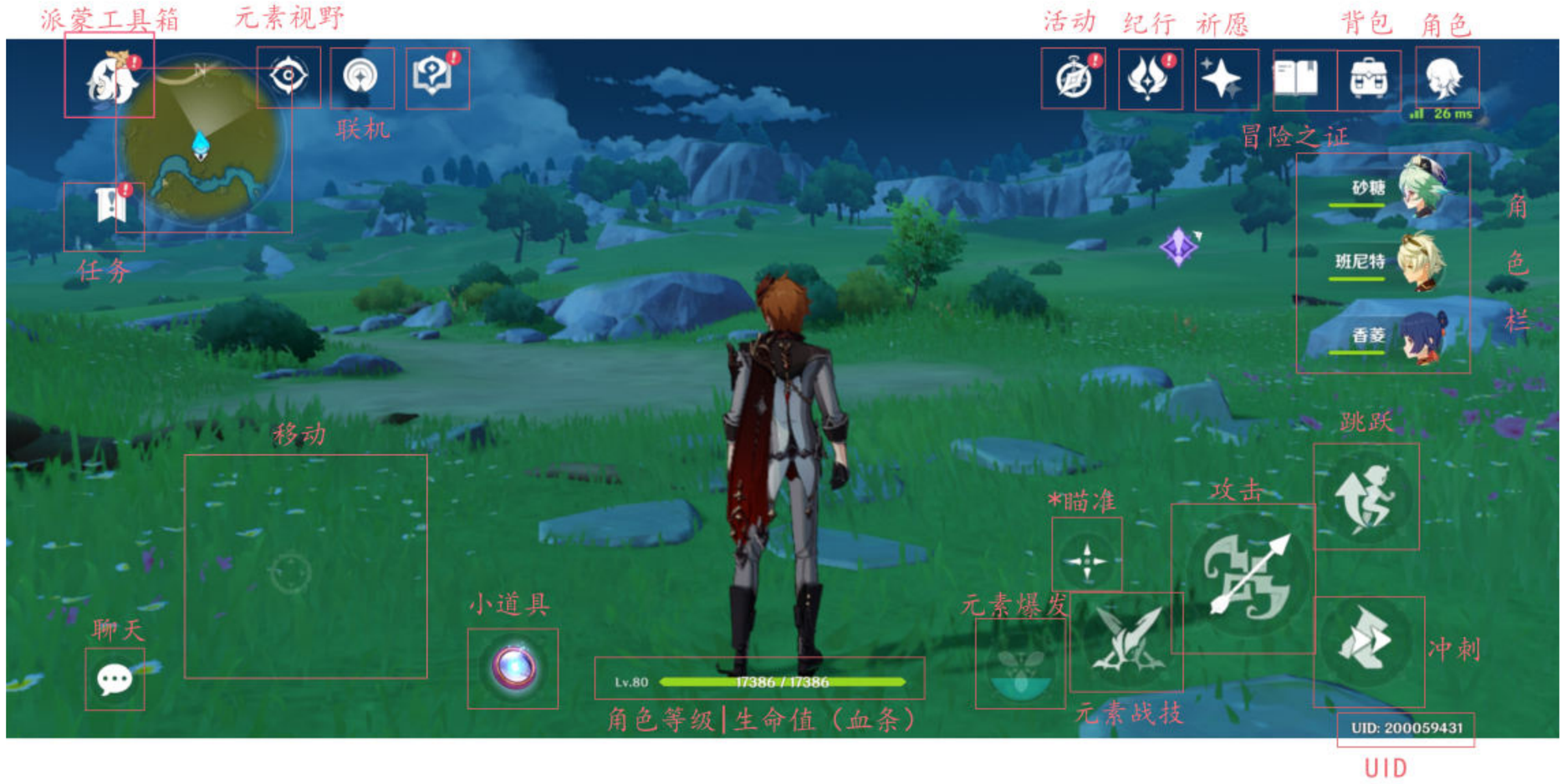
继承

- 类B继承自类A：
  - 保留/可访问A的（某些）数据/方法
  - B可以有自己的数据和方法，B既是B类型，也同时是A类型的

猫科

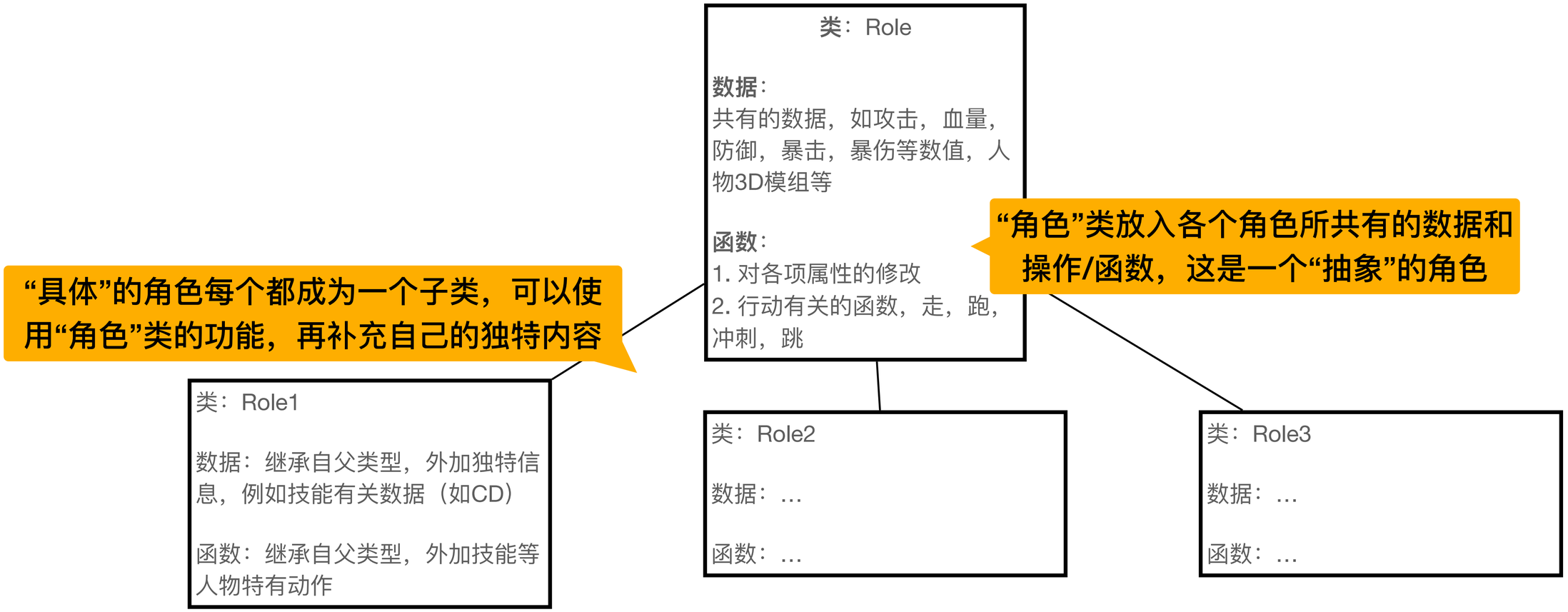

猫科 Felidae
豹世系
豹亚科 Pantherinae
豹属 Panthera
豹 (P. pardus)
狮 (P. leo)
美洲豹 (P. onca)
雪豹 (P. uncia)
虎 (P. tigris)
云豹属 Neofelis
云豹 (N. nebulosa)
巽他云豹 (N. diardi)
猫亚科 Felinae
金猫世系
金猫属 Catopuma
婆罗洲金猫 (C. badia)
金猫 (C. temminckii)
纹猫属 Pardofelis
纹猫 (P. marmorata)
狞猫世系
狞猫属 Caracal
狞猫 (C. caracal)
非洲金猫 (C. aurata)
薮猫属 Leptailurus
薮猫 (L. serval)
虎猫世系
虎猫属 Leopardus
乔氏猫 (L. geoffroyi)
南美林猫 (L. guigna)
小斑虎猫 (L. tigrina)
山原猫 (L. jacobita)
南美草原猫 (L. colocola)
虎猫 (L. pardalis)
长尾虎猫 (L. wiedii)
猞猁世系
猞猁属 Lynx
猞猁 (L. lynx)
伊比利亚猞猁 (L. pardinus)
加拿大猞猁 (L. canadensis)
短尾猫 (L. rufus)
美洲狮世系
美洲金猫属 Puma
美洲狮 (P. concolor)
细腰猫属 Herpailurus
细腰猫 (H. yagouaroundi)
猎豹属 Acinonyx
猎豹 (A. jubatus)
豹猫世系
豹猫属 Prionailurus
豹猫 (P. bengalensis)
渔猫 (P. viverrinus)
扁头豹猫 (P. planiceps)
锈斑豹猫 (P. rubiginosus)
兔狲属 Otocolobus
兔狲 (O. manul)
猫属 Felis
丛林猫 (F. chaus)
黑足猫 (F. nigripes)
沙漠猫 (F. margarita)
斑猫
荒漠猫 (F. bieti)
亚非野猫 (F. lybica)
欧洲野猫 (F. silvestris)
家猫 (F. catus)
家猫世系

以《原神》角色系统为例谈继承



以《原神》角色系统为例谈继承

- 希望对角色系统进行建模



函数重写：子类重新实现父类的相同函数

Function override

https://www.bilibili.com/video/BV1rP4y147Fi

类：Role

数据：
共有的数据，如攻击，血量，防御，暴击，暴伤等数值，人物3D模组等

函数：
1. 对各项属性的修改
2. 行动有关的函数，走，跑，冲刺，跳

个别角色的“跑”“冲刺”是特别技能

类：Role1

数据：继承自父类型，外加独特信息，例如技能有关数据（如CD）

函数：继承自父类型，外加技能等人物特有动作

类：Role2

数据：…

函数：…

类：SpecialRole1

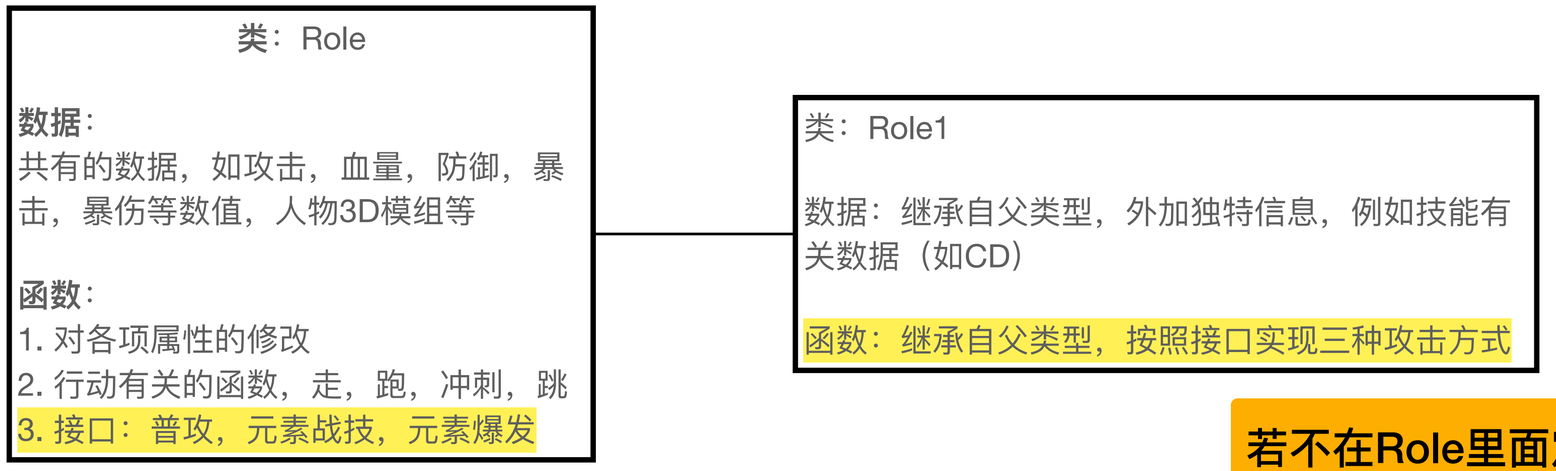
数据：…

函数：重写“跑”函数

function override

提供接口

- 接口类：规定若干在子类中必须要实现的操作/函数
- 外界只要知道某个类是一个接口A的子类，就一定可以调用这些接口方法



若不在Role里面定义接口是无法对Role类型调用“普攻”的

- 例如，设有Role A代表某个角色，则接收到鼠标左键点击，就调用A.普攻()

接口举例

- 例如定义一个Printable的类/接口，里面仅仅要求提供一个

String print()

即返回一个字符串，内容是当前对象的“可打印”内容

- 此时对于一切继承/实现了Printable类/接口的对象X，都可以调用X.print()来得到X内容的字符串表示

Java中各种地方都可以调用的toString函数

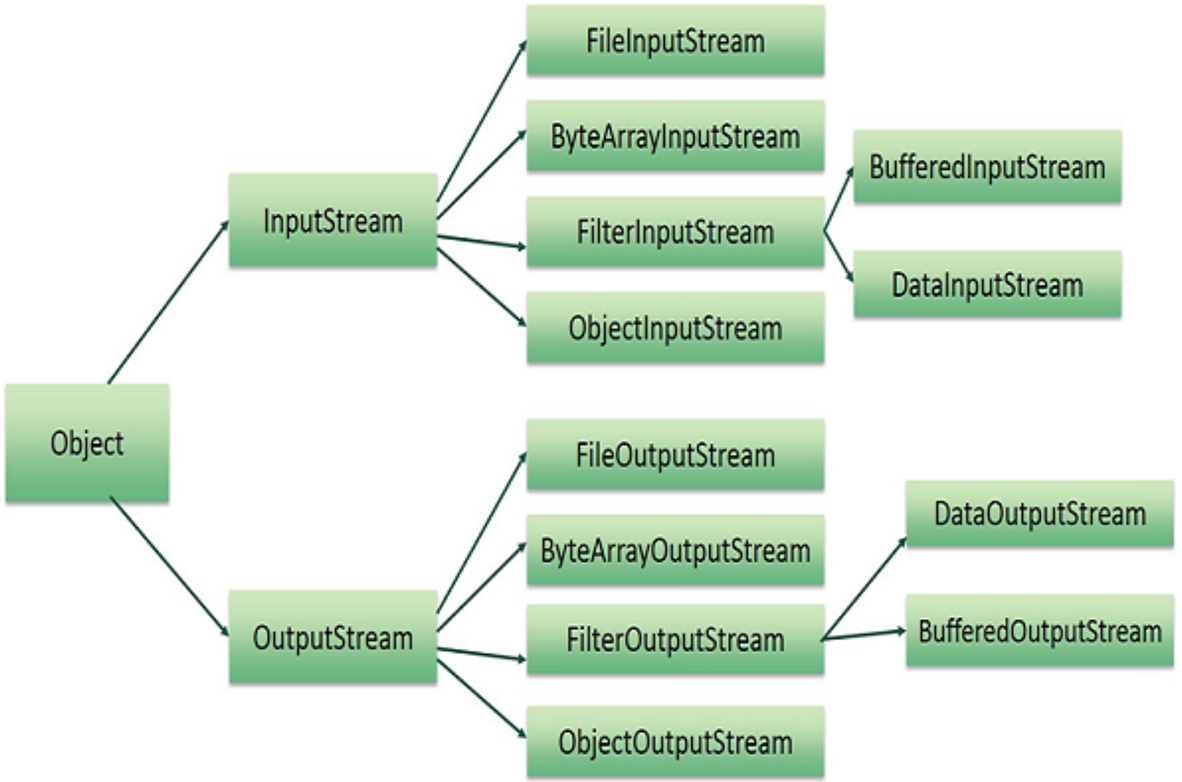
接口举例

- 又例如在IO库中，定义Input接口，要实现scanf函数
- 若干子类都继承自Input，然后实现scanf，具体实现取决于具体情况
  - 从压缩文件读取
  - 从stdio读取
  - 从文件读取
  - 从网络URL读取…

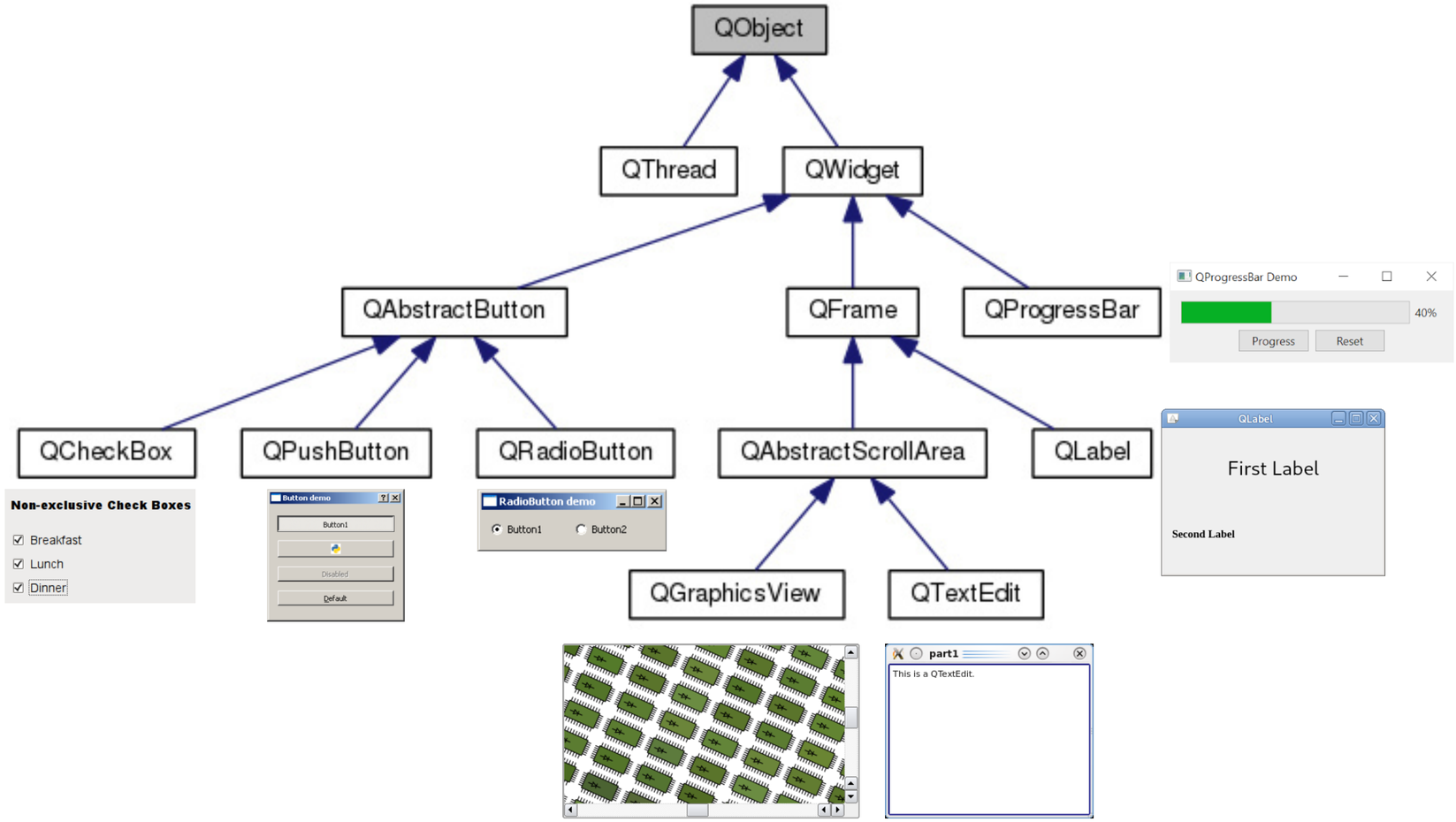
接口的好处

- 解耦合：接口可将功能定义与具体实现相分离
  - 依赖接口的算法不需要知道接口具体怎么实现的
  - 实现接口的类也不需要知道接口是如何被调用的
- 提高了可扩展性和可复用性：
  - 实现功能/类库的时候，可以把数据需要具备的最低限度的函数定义成接口
  - 以后有人想利用这些功能/类库的时候，只需要实现接口即可

例子：Java IO


Object
InputStream
FileInputStream
ByteArrayInputStream
FilterInputStream
ObjectInputStream
BufferedInputStream
DataInputStream
OutputStream
FileOutputStream
ByteArrayOutputStream
FilterOutputStream
ObjectOutputStream
DataOutputStream
BufferedOutputStream

例子：Qt GUI


QObject
QThread
QWidget
QAbstractButton
QFrame
QProgressBar
QCheckBox
QPushButton
QRadioButton
QAbstractScrollArea
QLabel
QGraphicsView
QTextEdit
QProgressBar Demo
40%
Progress
Reset
QLabel
First Label
Second Label
Non-exclusive Check Boxes
Breakfast
Lunch
Dinner
Button demo
Button1
Disabled
Default
RadioButton demo
Button1
Button2
part1
This is a QTextEdit.

面向过程就一定不好/不行吗?

- Linux kernel
  - 绝大多数C写成（少数其他语言，如汇编）
  - 几千万行代码，源码压缩包100M+（tar xz）
  - https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/fs/ext4/ext4.h?h=v6.2.8
- Tex: http://mirrors.ctan.org/systems/knuth/dist/tex/tex.web

大作业

- http://cssyb.openjudge.cn/23bighw/
- Deadline: 6月1日
- 助教会检查是否使用面向对象编程
  - 若未使用则分数 * 0.8（使用则分数不变）
  - 什么叫做使用：至少类与继承都使用过一次
- 注意养成良好的代码风格 (可考虑借助lint)