Object-Oriented Programming (OOP)

Week – 08

Oct 29, 2020

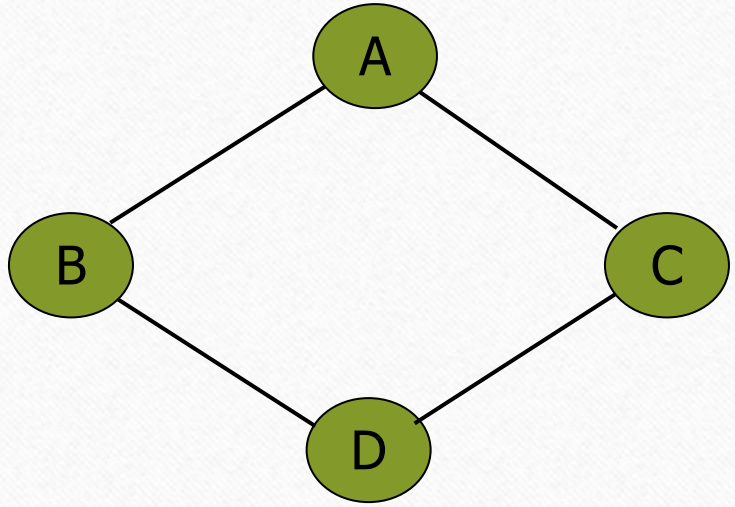
Instructor: **Talha Khan**

Email: talha.khan@nu.edu.pk

Object-Oriented Programming (OOP)

Hybrid Inheritance: Potential problem

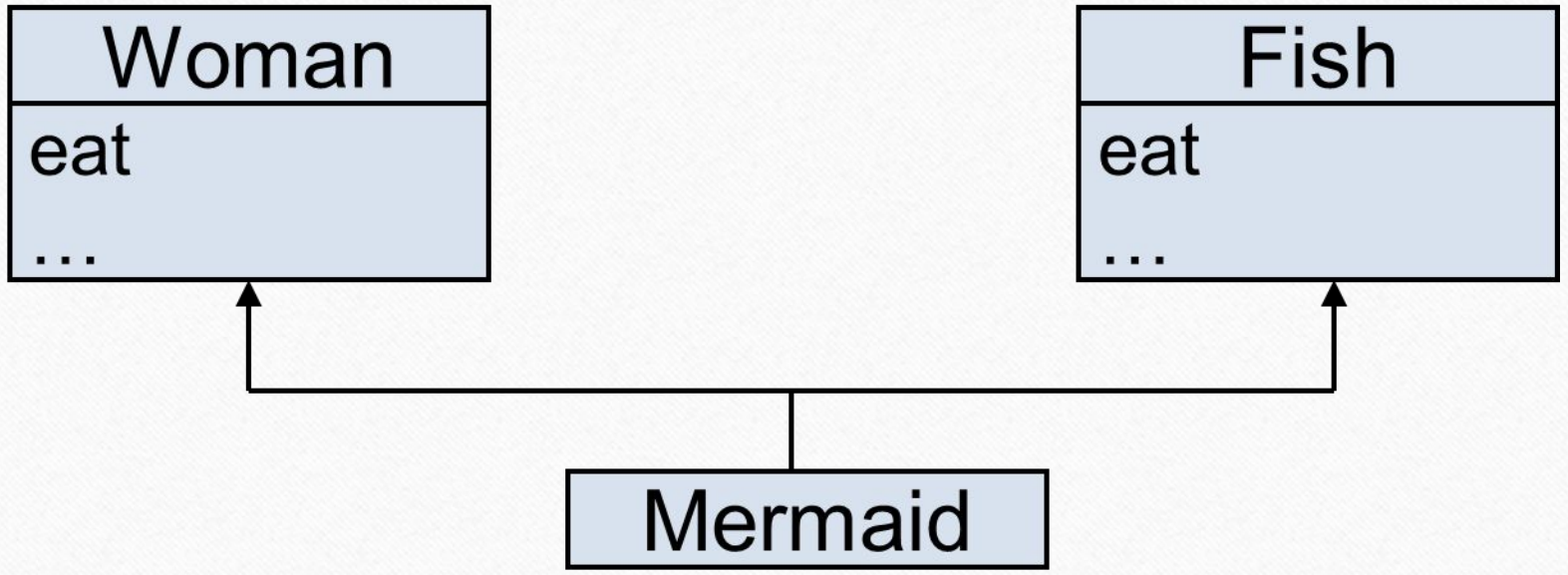
- common dangerous pattern: "The Diamond"
  - Classes B and C extend A
  - Class D extends A and B
  - Class D extends A and C



Problems with Hybrid Inheritance

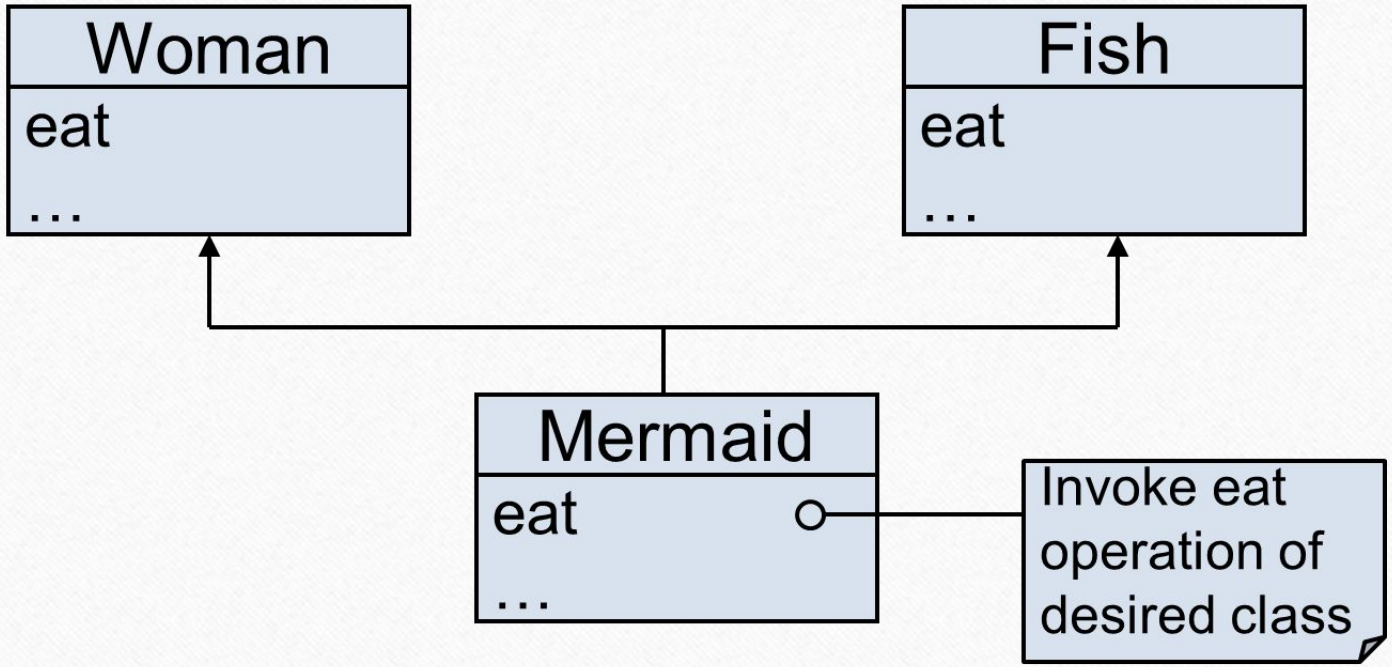
- Increased complexity
- Reduced understanding
- Duplicate features

Problem – Duplicate Features

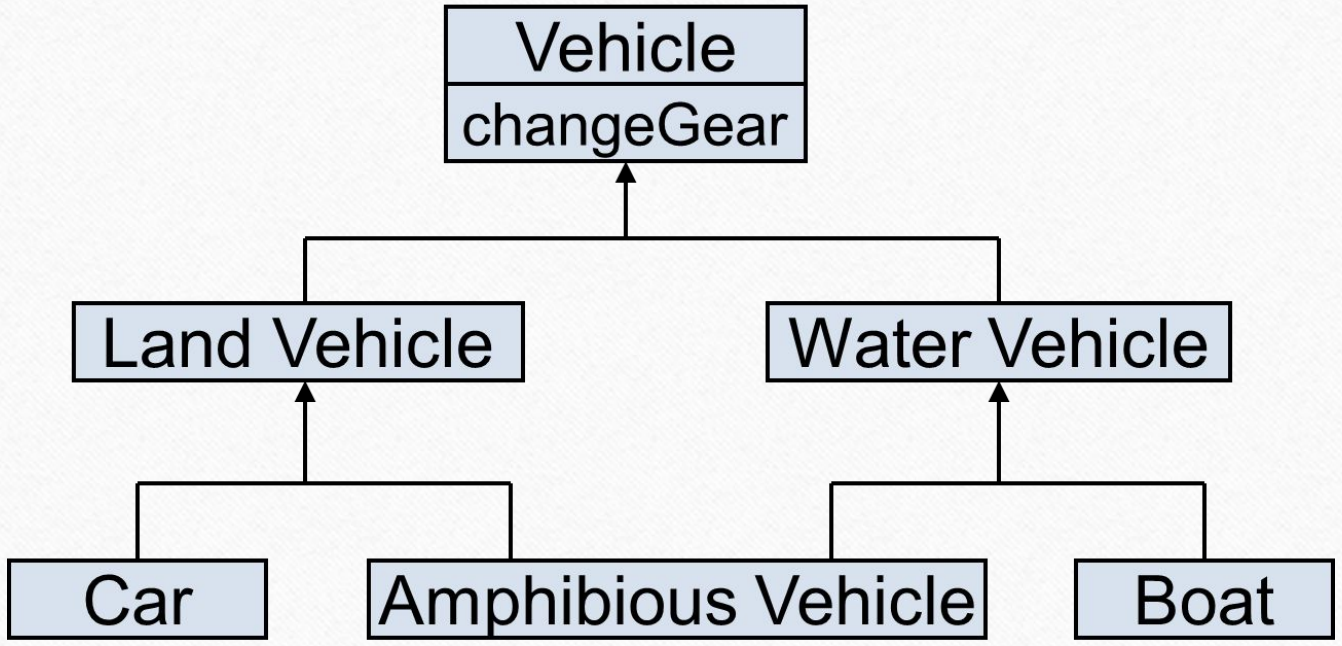


Which *eat* operation *Mermaid* inherits?

Solution – Override the Common Feature



Problem – Duplicate Features (Diamond Problem)



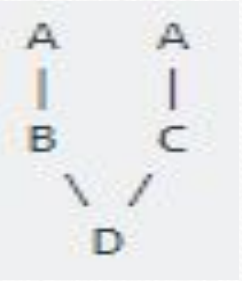
Which *changeGear* operation Amphibious Vehicle inherits?

Solution to Diamond Problem

- Some languages disallow diamond hierarchy
- Others provide mechanism to ignore characteristics from one side

Solution to Diamond Problem (Virtual Inheritance)

- What happens without virtual inheritance:



- You want: (Achievable with virtual inheritance)

Solution..

```
4  class LivingThing {
5
6  public:
7
8  void breathe()
9  {
10 cout << "I'm breathing as a living thing." <<endl;
11 }
12 };
```

```
33 int main()
34 {
35 Snake snake;
36 snake.breathe();
37 snake.crawl();
38 return 0;
39 }
```

```
14 class Animal : virtual public LivingThing {
15
16 public:
17 void breathe() {
18 cout << "I'm breathing as an animal." <<endl;
19 }
20 };
21
22 class Reptile : virtual public LivingThing {
23
24 public:
25
26 void crawl() {
27 cout << "I'm crawling as a reptile." <<endl;
28 }
29 };
30
31 class Snake :public Animal,public Reptile {};
32
```

Diamond Problem Solution (With constructor)

```
4  class Person {
5
6  public:
7      Person(int x)  {}
8      Person()       {}
9  };
```

```
28 int main()  {
29     TA ta1(30);
30 }
```

```
11 class Faculty : virtual public Person {
12
13 public:
14     Faculty(int x):Person(x)   {}
15 };
16
17 class Student : virtual public Person {
18
19 public:
20     Student(int x):Person(x) {}
21 };
22
23 class TA : public Faculty, public Student  {
24 public:
25     TA(int x):Student(x), Faculty(x), Person(x)   {}
26 };
```

Benefit of Virtual Inheritance

- In general, it is not allowed to call the grandparent's constructor directly, it has to be called through parent class. It is allowed only when 'virtual' keyword is used.

Calling Parameterized Constructor of Grandparent class

- In this case we can call the parameterized constructor of grandparent class directly through grandchild class.