

# Gondolatok a HF-hez

- Megírtuk a világ legjobb snake játékát funkcionálisan.
- OO modellre való áttérésnél kézenfekvő, hogy kígyó és gyümölcs is objektum.

```
// eredeti kódrészlet:  
void eat(snake s,  
        enum fruit f) {  
    switch(f) {  
        case apple:  
            s.length += 1;  
            break;  
        case kiwi:  
            s.length += 3;  
            break;  
        default:  
            }  
    }
```

```
// OO-nak GONDOLT változat  
enum kind { apple, pear, kiwi };  
class Fruit {  
    kind type;  
public:  
    kind getType() { return type; }  
};  
  
class Snake {  
    int length;  
public:  
    Snake() : length(1) {}  
    void grow(int n) { length += n; }  
    void feed(Fruit& f) {  
        switch(f.getType()) {  
            case apple:  
                grow(1);  
            break;  
            case kiwi:  
                ...  
        }  
    }  
};
```

Ez így nem OO program

# Mi a baj?

```
// OO-nak GONDOLT változat
enum kind { apple, pear, kiwi };
class Fruit {
    kind type;
public:
    kind getType() { return type; }
};

class Snake {
    int length;
public:
    Snake() : length(1) {}
    void grow(int n) { length += n; }
    void feed(Fruit& f) {
        switch(f.getType()) {
            case apple:
                grow(1);
                break;
            case kiwi:
                ...
        }
    }
};
```

Ez így nem OO program

- Ez C++ szintakszissal írt procedurális kód.
- SWITCH eleve gyanús!!!
- Ha típus kell lekérdezi az is nagyon gyanús!!!!
- Objektumok nem hatnak/kommunikálnak egymással.
- Nehezen módosítható. Módosítani kell a kígyó kódját pl:
  - Egy új gyümölcs megjelenése miatt.
  - Gyümölcstől függő méretű növekedést akarunk.
  - ...

# OO megvalósítás

- Módosítható
- Bővíthető
- Átlátható

```
class Fruit {
public:
    virtual void eatenBy(Snake* s);
};

class Snake {
    int length;
public:
    Snake() : length(1) {}
    void grow(int n) { length += n; }
    void feed(Fruit& f) {
        f.eatenBy(this);
    }
    void draw() ...
};

class Apple : public Fruit {
public:
    void eatenBy(Snake* s) {
        s->grow(1);
    }
};
```

# Mérgező alma is lehet

```
class Fruit {
public:
    virtual void eatenBy(Snake* s);
};

class Snake {
    int length;
    bool killed;
public:
    Snake() : length(1),
             killed(false) {}
    void grow(int n) { length += n; }
    void kill() { killed = true; }
    void feed(Fruit& f) {
        f.eatenBy(this);
    }
    void draw() ...
};
```

```
class PoisonApple :
    public apple {
public:
    void eatenBy(Snake* s) {
        s->kill();
    }
};
```