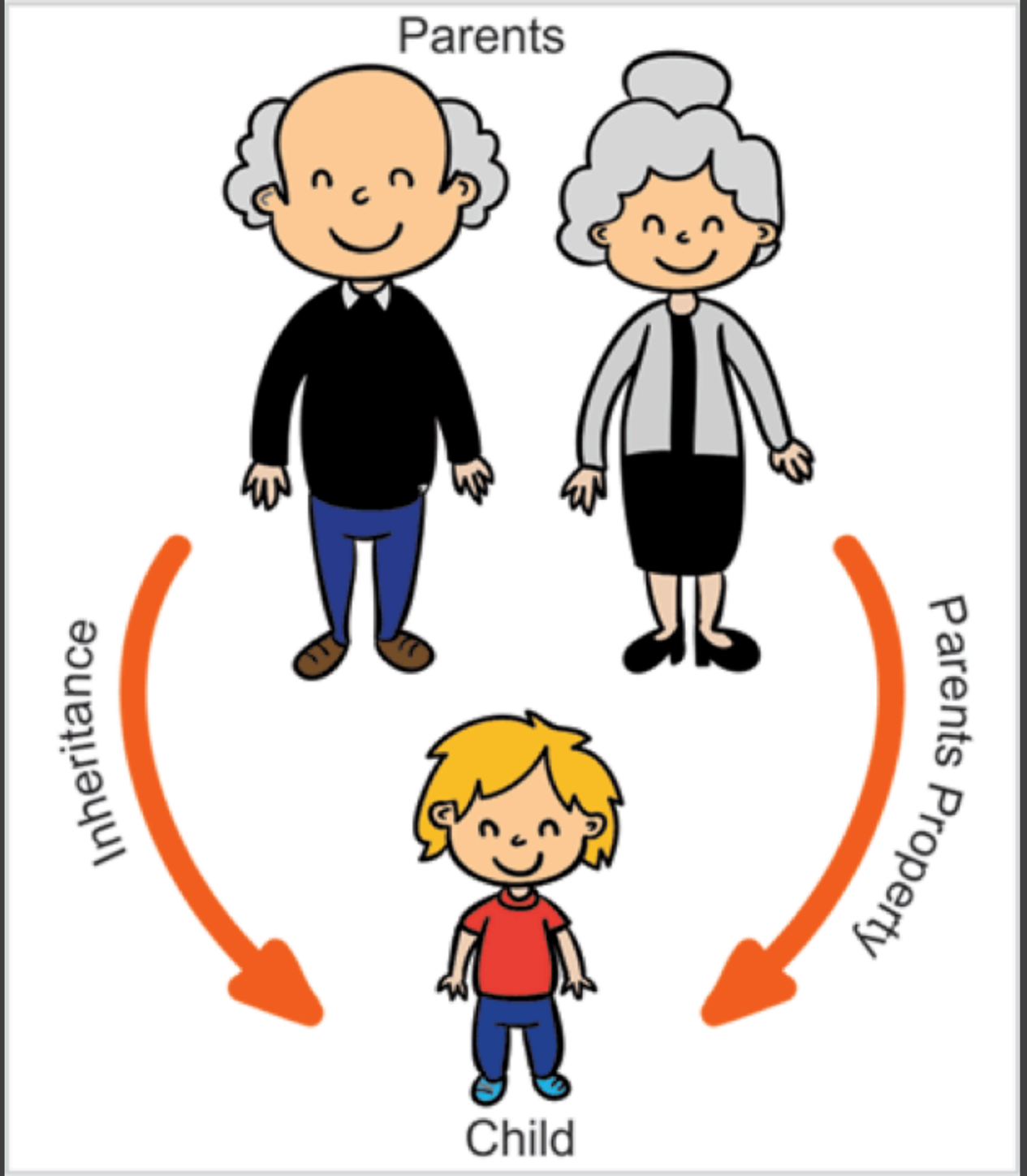
Objektinis Programavimas

Paveldėjimas (*Inheritance*)



Turinys

1. Motyvacija
2. Paveldėjimas
   <!-- 3. Išvestinės klasės konstruktoriai -->

Motyvacija (1)

Kelios kodo eilutės, kelios sugaištos minutės ir 4-a praktinė užduotis "*atlikta*"! 😄

```
#include <iostream>
#include <vector>

template<typename T>
class Vector : public std::vector<T> {
public:
    using std::vector<T>::vector; // naudoti c-tor'ius iš std::vector
};

int main() {
    Vector<int> v(10,1);
    std::cout << "v[9] = " << v[9] << std::endl;
    v.push_back(2);
    std::cout << "v[10] = " << v[10] << std::endl;
    std::cout << "v[11] = " << v[11] << std::endl; // Ką gausime čia?
}
```

Motyvacija (2)

Norint prisitaikyti egzistuojančių klasių funkcionalumą savo poreikiams

```
#include <iostream>
#include <vector>

template<typename T>
class Vector : public std::vector<T> {
public:
    using std::vector<T>::vector;                          // naudoti c-tor'ius iš std::vector
    T& operator[](int i) { return std::vector<T>::at(i); } // patikrina range
    const T& operator[](int i) const { return std::vector<T>::at(i); } // patikrina range const objektams
};

int main() {
    Vector<int> v(10,1);
    std::cout << "v[9] = " << v[9] << std::endl;
    v.push_back(2);
    std::cout << "v[10] = " << v[10] << std::endl;
    std::cout << "v[11] = " << v[11] << std::endl; // O ką dabar gausime?
}
```

Motyvacija (3)

Norint išplėsti egzistuojančių klasių funkcionalumą

```
/* Vector kodas iš ankstesnės skaidrės */
// Vector'ių sudėtis
template<typename T>
Vector<T> operator+(const Vector<T>& a, const Vector<T>& b) {
  if (a.size() != b.size())
    throw std::runtime_error("Vektorių dydžio neatitikimas!");
  auto size = a.size();
  Vector<T> c(size);
  for (size_t i = 0; i != a.size(); ++i)
    c[i] = a[i] + b[i];
  return c;
}

int main() {
    Vector<int> v1(10,1);
    Vector<int> v2(10,2);
    Vector<int> v3 = v1 + v2; // sudėdame Vector'ius
}
```

Paveldėjimas (*inheritance*) (1)

— Objektiškai orientuotame programavime (OOP) **paveldėjimas** (*inheritance*) yra vienas svarbiausių ir naudingiausiųs principų/mechanizmų.

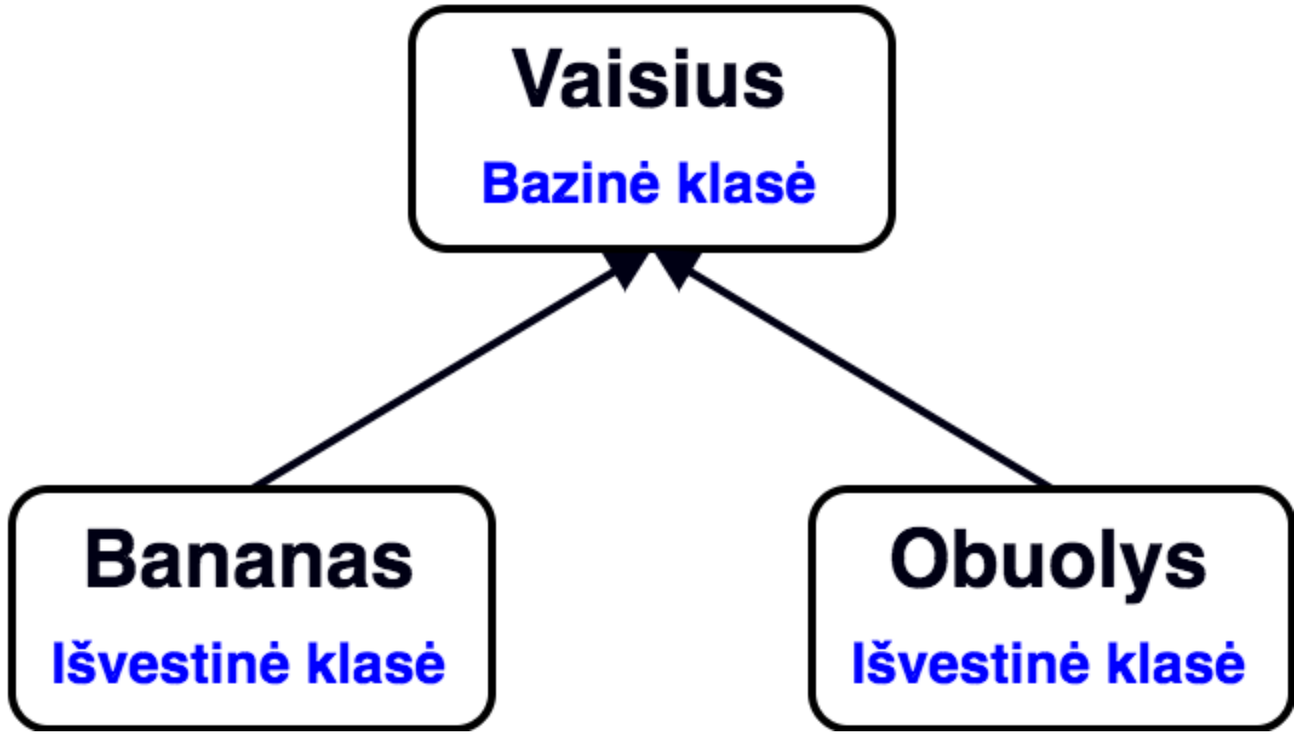
— C++ kalboje klasės gali paveldėti kitos klasės ar net kelių klasių duomenis (*member variables*) ir metodus (*member functions*).

— Tokiu būdu galime kurti naujas klases modifikuojant ir/ar išplečiant egzistuojančių funkcionalumą.

Paveldėjimas (*inheritance*) (2)

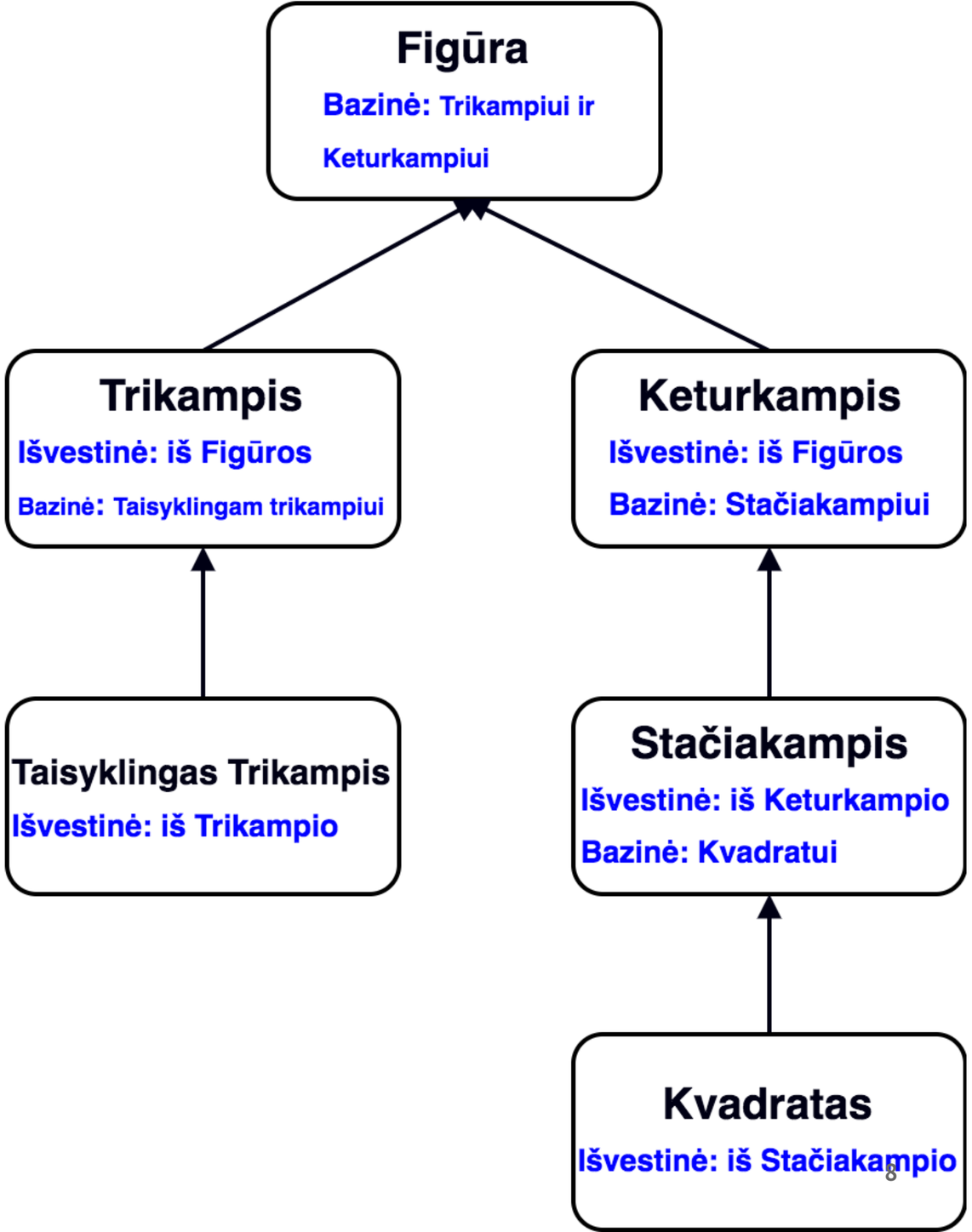
— Klasė, iš kurios paveldimos visos savybes vadinama **bazinė klasė** (*base class*) ar (*parent class*), o ją papildanti - **išvestinė klasė** (*derived class*) ar (*child class*).

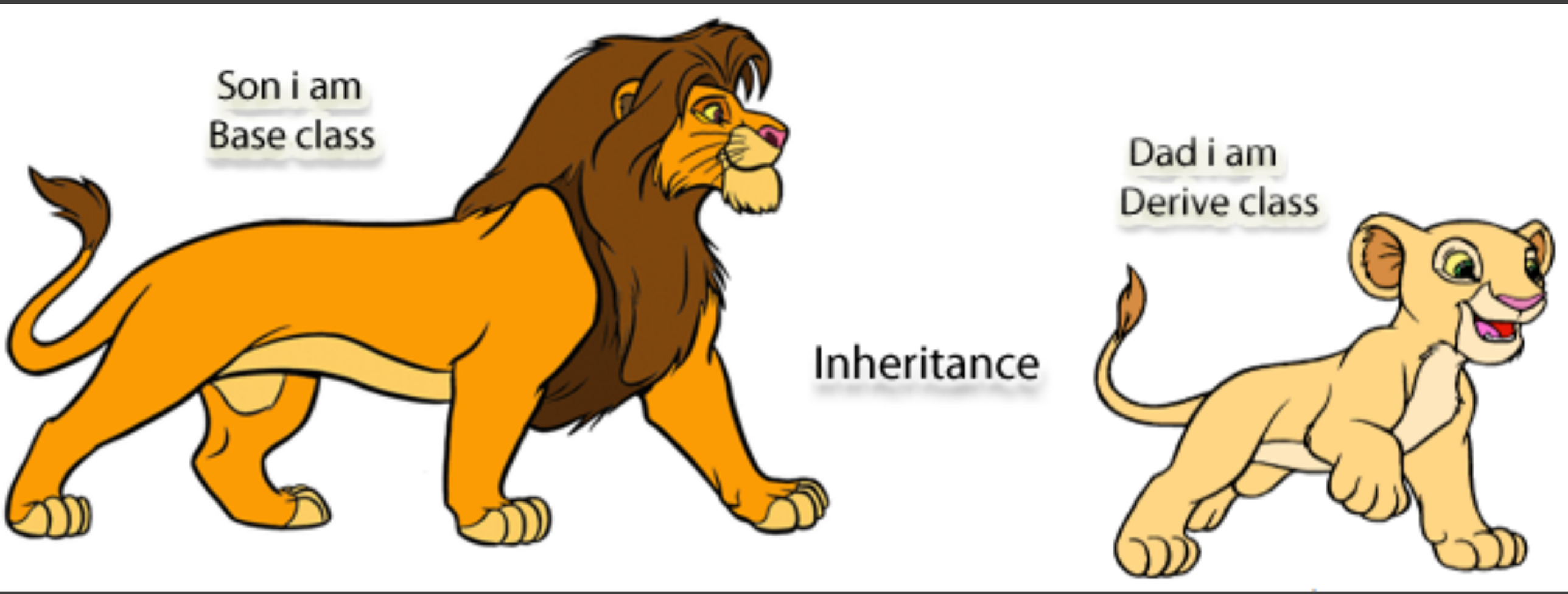
— Jei bazinėje klasėje ištaisomos klaidos ar realizuojamos naujos funkcijos, visa tai automatiškai paveldi išvestinės klasės!



Paveldėjimo grandinės

— Tam tikros klasės tuo pat metu gali būti išvestinės (iš ankstesnės bazinės klasės), bet ir pačios yra bazinės klasės iš jų išvedamoms klasėms.




Son i am
Base class
Inheritance
Dad i am
Derive class