Πανεπιστήμιο Αιγαίου

Μεθοδολογίες και Γλώσσες Προγραμματισμού Ι

Κληρονομικότητα

Εργίνα Καβαλλιεράτου (kavallieratou@aegean.gr)

Μόνιμη Επίκουρος Καθηγήτρια

Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σήμερα!

- ✓ Κλάση Βάσης
- ✓ Παράγωγη κλάση
- ✓ Απλή κληρονομικότητα
- ✓ Protected δεδομένα
- ✓ Constructors & Destructors
- ✓ overloading

Κλάση Βάση/Παράγωγη

- ✓ Τα διάφορα αντικείμενα μπορούν να έχουν μεταξύ τους σχέση ταξινόμησης π.χ Θηλαστικό - Σκύλος
- ✓ Η C++ δίνει τη δυνατότητα απεικόνισης τέτοιων σχέσεων, ορίζοντας κλάσεις παραγόμενες από άλλες.
- ✓ Για παράδειγμα μπορούμε να ορίσουμε την κλάση Σκύλος ως παράγωγη της κλάσης Θηλαστικό.
- ✓ Σε αυτή την περίπτωση π.χ δεν χρειάζεται να ορίσουμε ότι ο Σκύλος κινείται αν έχουμε ήδη ορίσει ότι το Θηλαστικό κινείται.
- ✓ Η κλάση Θηλαστικό λέγεται Βάση και η Σκύλος Παράγωγη.
- ✓ Μία κλάση Βάση μπορεί να έχει πολλές Παράγωγες.

Σχέσεις "IS-A", "HAS-A"

- ✓ **Σχέση "IS-A"**
  - **"IS-A", υποδηλώνει σχέση κληρονομικότητας**
    - Ένα αντικείμενο μιας παραγόμενης κλάσης μπορούμε να το χειριστούμε και ως αντικείμενο της κλάσης βάσης
    - Παράδειγμα: *Αυτοκίνητο, Φορτηγό, Μοτοσικλέτα* **IS-A** *Όχημα*
      - Οι ιδιότητες/συμπεριφορά της κλάσης Όχημα ισχύουν και για την κλάση Αυτοκίνητο, Φορτηγό, Μοτοσικλέτα
- ✓ **Σχέση "HAS-A"**
  - **"HAS-A", υποδηλώνει σχέση σύνθεσης**
    - Ένα αντικείμενο περιέχει ένα ή περισσότερα αντικείμενα άλλων κλάσεων ως μέλη
    - Παράδειγμα: *Αυτοκίνητο* **HAS-A** 1 *Μηχανή*, 1 *Τιμόνι*, 4 *Πόρτες*, κτλ

Παραδείγματα Κληρονομικότητας

Κλάση Βάσης	Κλάση Παράγωγη
Υπάλληλος	Πλήρους Απασχόλησης Μερικής Απασχόλησης
Δάνειο	Φοιτητικό Καταναλωτικό Στεγαστικό
Σχήμα	Κύκλος Τρίγωνο Ορθογώνιο
Φοιτητής	Μεταπτυχιακός Προπτυχιακός
Λογαριασμός	Όψεως Ταμιευτηρίου

Παραγόμενες κλάσεις

- ✓ Ας φανταστούμε ότι πρέπει να φτιάξουμε ένα πρόγραμμα που περιγράφει μία φάρμα ζώων.
- ✓ Θα πρέπει να συμπεριλάβουμε κλάσεις για τα διάφορα ζώα.
- ✓ Όταν δηλώνουμε μία κλάση θα πρέπει να υποδεικνύουμε από ποια κλάση παράγεται, γράφοντας μετά το όνομα της κλάσης, τον τρόπο παραγωγής:

```
class Dog : public Mammal
```

- ✓ Η κλάση Βάση **ΠΡΕΠΕΙ** να έχει δηλωθεί νωρίτερα.

Απλή κληρονομικότητα/1

```
#include <iostream.h>
enum BREED { YORKIE, CAIRN,DANDIE,SHETLAND,DOBERMAN,LAB };
class Mammal    {
public:
   Mammal();
   ~Mammal();
   int GetAge()const;
   void SetAge(int);
   int GetWeight() const;
   void SetWeight();
   void Speak();
   void Sleep();
protected:
   int itsAge;
   int itsWeight;  };
```

Απλή κληρονομικότητα/2

```
class Dog : public Mammal
{
public:
   Dog();
   ~Dog();
   BREED GetBreed() const;
   void SetBreed(BREED);
   WagTail();
   BegForFood();

protected:
   BREED itsBreed;
};
```

Protected δεδομένα

- ✓ Η παραγώμενη κλάση, κληρονομεί από την κλάση βάσης όλα τα δεδομένα και τις συναρτήσεις, **εκτός** από τον τελεστή αντιγραφής, το δομητή και τον αποδομητή.
- ✓ Τα private μέλη δεν είναι διαθέσιμα στις παραγόμενες κλάσεις
- ✓ Τα protected μέλη είναι απολύτως διαθέσιμα στις παραγόμενες κλάσεις και private για το υπόλοιπο πρόγραμμα.
- ✓ Οι συναρτήσεις μέλη έχουν πρόσβαση σε όλα τα δεδομένα και συναρτήσεις της κλάσης τους (ακόμα και τα private) και στα public και protected μέλη των κλάσεων βάσης.

Protected δεδομένα

Ο παρακάτω πίνακας συνοψίζει τις δυνατότητες πρόσβασης στα μέλη της κλάσης βάσης και της παράγωγης κλάσης (κληρονομικότητα με δημόσια-public πρόσβαση)

Μέλη κλάσης βάσης	Πρόσβαση από αντικείμενα της κλάσης βάσης	Πρόσβαση από την παράγωγη κλάση	Πρόσβαση από αντικείμενα της παράγωγης κλάσης
public	ΝΑΙ	ΝΑΙ	ΝΑΙ
private	ΟΧΙ	ΟΧΙ	ΟΧΙ
protected	ΟΧΙ	ΝΑΙ	ΟΧΙ

Παραγόμενο αντικείμενο/1

```
#include <iostream.h>
enum BREED{YORKIE,CAIRN,DANDIE,SHETLAND,DOBERMAN,LAB};
class Mammal  {
public:
  Mammal():itsAge(2), itsWeight(5){}
  ~Mammal(){}
  int GetAge()const { return itsAge; }
  void SetAge(int age) { itsAge = age; }
  int GetWeight() const { return itsWeight; }
  void SetWeight(int weight) { itsWeight = weight; }
  void Speak()const { cout << "Mammal sound!\n"; }
  void Sleep()const { cout << "shhh. I'm sleeping.\n"; }
protected:
  int itsAge;
  int itsWeight;  };
```

Παραγόμενο αντικείμενο/2

```
class Dog : public Mammal
{
public:
  Dog():itsBreed(YORKIE){}
  ~Dog(){}
  BREED GetBreed() const { return itsBreed; }
  void SetBreed(BREED breed) { itsBreed = breed; }
  void WagTail() { cout << "Tail wagging...\n"; }
  void BegForFood() { cout << "Begging for food...\n"; }
private:
  BREED itsBreed;
};
```

Παραγόμενο αντικείμενο/1

```
int main()
{
   Dog fido;
   fido.Speak();
   fido.WagTail();
   cout << "Fido is " << fido.GetAge() << " years old\n";
 return 0;
}
```

```
Mammal sound!
Tail wagging...
Fido is 2 years old
```

Constructors & Destructors

- ✓ Ένα Dog αντικείμενο είναι και αντικείμενο Mammal.
- ✓ Όταν δημιουργείται το Fido, πρώτα καλείται ο δομητής της βάσης και μετά του Dog.
- ✓ Όταν το Fido καταστρέφεται πρώτα καλείται ο αποδομητής του Dog και μετά του Mammal.

Constructors & Destructors/1

```
#include <iostream.h>
enum BREED{YORKIE,CAIRN,DANDIE,SHETLAND,DOBERMAN,LAB};
class Mammal  {
public:
  Mammal();
  ~Mammal();
  int GetAge()const { return itsAge; }
  void SetAge(int age) { itsAge = age; }
  int GetWeight() const { return itsWeight; }
  void SetWeight(int weight) { itsWeight = weight; }
  void Speak()const { cout << "Mammal sound!\n"; }
  void Sleep()const { cout << "shhh. I'm sleeping.\n"; }
protected:
  int itsAge;
  int itsWeight;  };
```

Constructors & Destructors/2

```
class Dog : public Mammal
{
public:
   Dog();
   ~Dog();
   BREED GetBreed() const { return itsBreed; }
   void SetBreed(BREED breed) { itsBreed = breed; }
   void WagTail() { cout << "Tail wagging...\n"; }
   void BegForFood() { cout << "Begging for food...\n"; }
private:
   BREED itsBreed;
};
```

Constructors & Destructors/3

```
Mammal::Mammal(): itsAge(1), itsWeight(5) {
     cout << "Mammal constructor...\n"; }
Mammal::~Mammal()  {
     cout << "Mammal destructor...\n"; }
Dog::Dog():  itsBreed(YORKIE)  {
    cout << "Dog constructor...\n"; }
Dog::~Dog() {
    cout << "Dog destructor...\n";  }
int main()   {
  Dog fido;
  fido.Speak();
  fido.WagTail();
  cout << "Fido is " << fido.GetAge() << " years old\n";
 return 0; }
```

```
Mammal constructor...
Dog constructor...
Mammal sound!
Tail wagging...
Fido is 1 years old
Dog destructor...
Mammal destructor...
```

Constructor overloading/1

```
#include <iostream.h>
enum BREED{YORKIE,CAIRN,DANDIE,SHETLAND,DOBERMAN,LAB};
class Mammal  {
 public:
   Mammal();
   Mammal(int age);
   ~Mammal();
   int GetAge()const { return itsAge; }
   void SetAge(int age) { itsAge = age; }
   int GetWeight() const { return itsWeight; }
   void SetWeight(int weight) { itsWeight = weight; }
   void Speak()const { cout << "Mammal sound!\n"; }
   void Sleep()const { cout << "shhh. I'm sleeping.\n"; }
protected:
   int itsAge;
   int itsWeight;  };
```

Constructor overloading/2

```
class Dog : public Mammal    {
public:
   Dog();
   Dog(int age);
   Dog(int age, int weight);
   Dog(int age, BREED breed);
   Dog(int age, int weight, BREED breed);
   ~Dog();
   BREED GetBreed() const { return itsBreed; }
   void SetBreed(BREED breed) { itsBreed = breed; }
   void WagTail() { cout << "Tail wagging...\n"; }
   void BegForFood() { cout << "Begging for food...\n"; }
private:
   BREED itsBreed;  };
```

Constructor overloading/3

```
Mammal::Mammal():  itsAge(1),  itsWeight(5)  {
   cout << "Mammal constructor...\n";  }
Mammal::Mammal(int age):  itsAge(age),  itsWeight(5)  {
   cout << "Mammal(int) constructor...\n";  }
Mammal::~Mammal()  {
   cout << "Mammal destructor...\n";  }
 Dog::Dog(): Mammal(), itsBreed(YORKIE)  {
   cout << "Dog constructor...\n";  }
 Dog::Dog(int age): Mammal(age), itsBreed(YORKIE) {
   cout << "Dog(int) constructor...\n"; }
 Dog::Dog(int age, int weight):  Mammal(age), itsBreed(YORKIE) {
   itsWeight = weight;
   cout << "Dog(int, int) constructor...\n";  }
```

Constructor overloading/3

```
Dog::Dog(int age, int weight, BREED breed): Mammal(age), itsBreed(breed){
      itsWeight = weight;
      cout << "Dog(int, int, BREED) constructor...\n";  }
Dog::Dog(int age, BREED breed):  Mammal(age), itsBreed(breed)  {
     cout << "Dog(int, BREED) constructor...\n";   }
Dog::~Dog()  {
     cout << "Dog destructor...\n";  }
```

Constructor overloading/4

```
int main()
 {
    Dog fido;
    Dog rover(5);
    Dog buster(6,8);
    Dog yorkie (3,YORKIE);
    Dog dobbie (4,20,DOBERMAN);
    fido.Speak();
    rover.WagTail();
    cout << "Yorkie is " << yorkie.GetAge()
                                  << " years old\n";
     cout << "Dobbie weighs ";
     cout << dobbie.GetWeight() << "
                                  pounds\n";
    return 0;
}
```

```
Mammal constructor...
Dog constructor...
Mammal(int) constructor...
Dog(int) constructor...
Mammal(int) constructor...
Dog(int, int) constructor...
Mammal(int) constructor...
Dog(int, BREED) constructor....
Mammal(int) constructor...
Dog(int, int, BREED) constructor...
Mammal sound!
Tail wagging...
Yorkie is 3 years old.
Dobbie weighs 20 pounds.
Dog destructor. . .
Mammal destructor...
Dog destructor...
Mammal destructor...
Dog destructor...
Mammal destructor...
Dog destructor...
Mammal destructor...
Dog destructor...
Mammal destructor...
```