AAB University

Faculty of Computer Sciences

Programimi i Orientuar në Objekte

Java 11:

Polimorfizmi

Doc. Dr. **Mentor Hamiti**

mentor.hamiti@universitetiaab.com

Programimi i Orientuar në Objekte

- Hyrje
- Klasa bazë dhe e trashëguar
- Polimorfizmi
- Pointerët në klasën bazë
- Anëtarët virtuel
- Klasat bazë abstrakte

Klasa bazë dhe e trashëguar

- Trashëgimia është koncept fundamental në gjuhët e orientuara në objekte
- Lejon mundësinë e krijimit të klasave të reja, të bazuara në ata paraprake
- Klasat e reja trashëgijnë të dhënat dhe fuksionet nga klasat bazë
  - Sjellja e klasës së re mund të modifikohet përmes funksioneve ekzistuese ose përmes shtimit të funksioneve të reja
  - Qasje kyqe në këtë rast paraqet **polymorfizmi** në ç'rast sjellja e klasës adaptohet gjatë egzekutimit (at run-time)

Klasa bazë dhe e trashëguar

- Egzistojnë shumë shembuj nga jeta reale, si klasa bazë (base) mund të ridefinohet në jë sërë klasash të derivuara (derived)

- Shembull:

  Shumëkëndëshi mund të ridefinohet në **Katërkëndësh**, i cili pastaj mund të modifikohet dhe të paraqet **Drejtkëndësh**

  - Shumëkëndëshi është Katërkëndësh
  - Katërkëndëshi është Drejtkëndësh

Klasa bazë dhe e trashëguar

- Shembull:

Base class	**Derived class**
Shape	Triangle, Circle, Rectangle
Bank Account	Current, Deposit
Student	Undergraduate, Postgaduate
Vehicle	Car, Truck, Bus
Filter	Low-pass, Band-pass, High-pass

Klasa bazë dhe e trashëguar

- Shembull: Klasa BankAccount

- Klasa BankAccount përmban të dhënat elementare për një acount banke:
  - *Account holder*
  - *Account number*
  - *Current balance*

- Fuksionet elementare:
  - *Withdraw money*
  - *Deposit money*

Class member	Can be accessed from
private	public member functions of same class
protected	public member functions of same class and derived classes
public	Anywhere

Polimorfizmi dhe POO

- **Polymorfizmi** gjithashtu është koncept fundamental në gjuhët e orientuara në objekte
- Polymorfizëm dtth shumë forma
- Ofron sjellje të ndryshme të objekteve nga një tip I vetëm referent

Polimorfizmi

- Mundësia e deklarimit të funksioneve si **virtuale** është element kyç i Polimorfizmit në **C++**

- Polimorfizmi: *nga Greqishtja*

  "të pasurit e më tepër formave të ndryshme"

  - *Funksioni i njejtë mund të thir objekte të ndryshme*
  - *Përgjigjet mund të jenë të ndryshme*

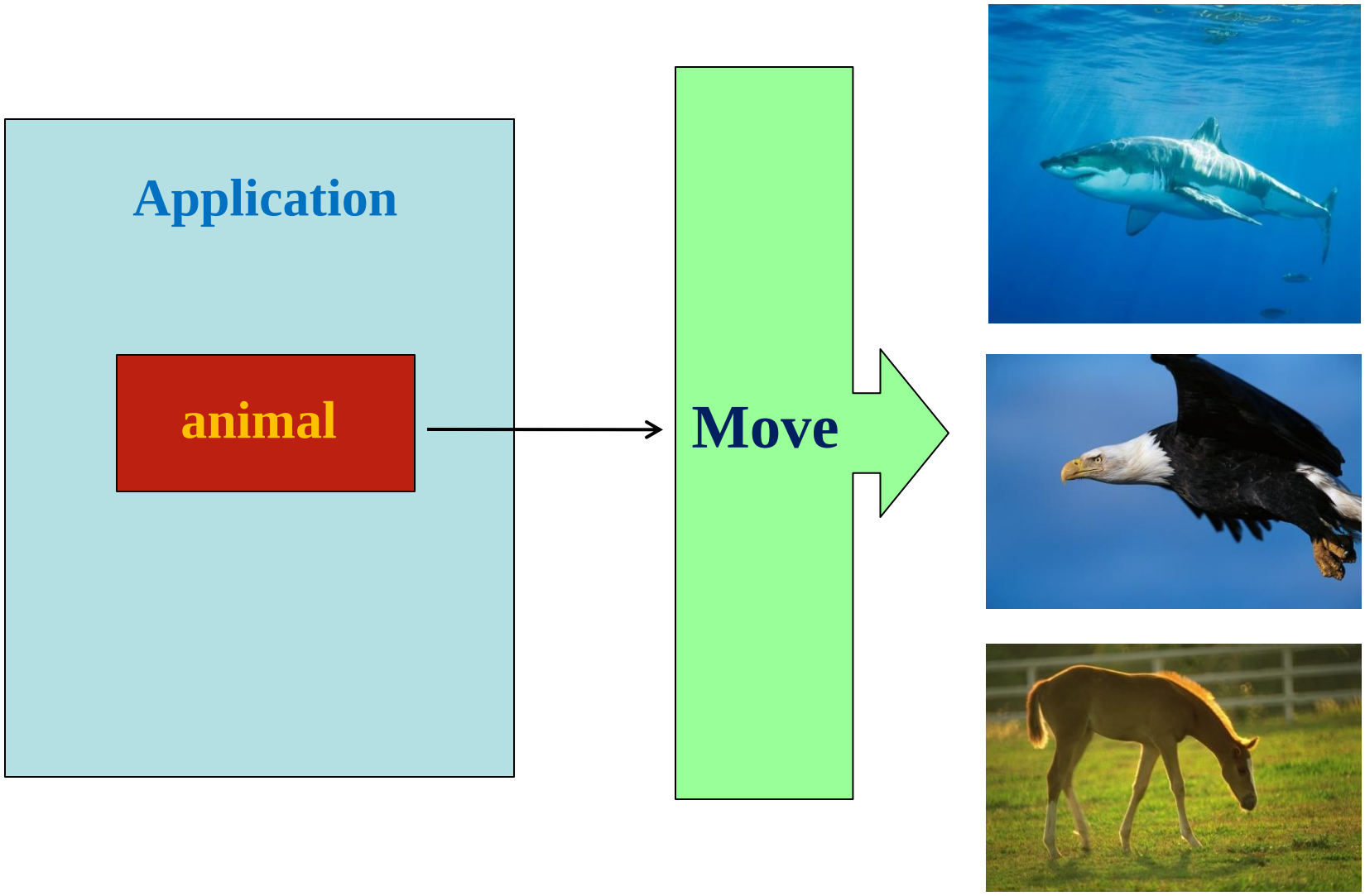
Polimorfizmi

- Shembull: Në lojrat kompjuterike që simulojnë lëvizjet e kafshëve shfrytëzohen urdhërat '**lëviz**' për lloje të ndryshme të **kafshëve**

- Urdhëri dërgohet përmes referencës në klasën **Kafshë**, e cila është klasë bazë për **kafshë** të ndryshme!

  - Në këtë rast, ç'do tip sillet ndryshe, pasi të ketë pranuar urdhërin
  - Kjo mundëson zgjerimin dhe mbindërtimin e aplikacionit

Polimorfizmi



Polimorfizmi

- **Polimorfizmi** implementohet përmes referencës në objekt

- Këta refernca mund tju ndahen objekteve të çdo klase të trashëguar

Pointerët në klasën bazë

- Karakteristikë me rëndësi e **trashëgimisë** është se **pointeri** në klasën e trashëguar është kompaktibil me pointerin në klasën bazë

- **Polimorfizmi** paraqet **art** të shfrytëzimit të këtyre veçorive pozzitive

Pointerët në klasën bazë

- Shembulli 1:

```
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b; }
};

class Rectangle: public Polygon {
  public:
    int area()
      { return width*height; }
};

class Triangle: public Polygon {
  public:
    int area()
      { return width*height/2; }
};
```

```
int main ()
{
  Rectangle rect;
  Triangle trgl;
  Polygon * ppoly1 = &rect;
  Polygon * ppoly2 = &trgl;
  ppoly1->set_values (4,5);
  ppoly2->set_values (4,5);
  cout << rect.area() << '\n';
  cout << trgl.area() << '\n';
  cin.get(); return 0;
}
```

```
20
10
```

Anëtarët virtuel

- Anëtarët **virtuel** paraqesin funksione që mund të ridefinohen në klasën e treshëguar, përmes thirjeve referente

- Syntaksa për një funksion që të deklarohet si virtuel është fjala kyqe **virtual**

- Shembulli 2:

```
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b; }
    virtual int area ()
      { return 0; }
};

class Rectangle: public Polygon {
  public:
    int area ()
      { return width * height; }
};

class Triangle: public Polygon {
  public:
    int area ()
      { return (width * height / 2); }
};
```

```
int main () {
  Rectangle rect;
  Triangle trgl;
  Polygon poly;
  Polygon * ppoly1 = &rect;
  Polygon * ppoly2 = &trgl;
  Polygon * ppoly3 = &poly;
  ppoly1->set_values (4,5);
  ppoly2->set_values (4,5);
  ppoly3->set_values (4,5);
  cout << ppoly1->area() << '\n';
  cout << ppoly2->area() << '\n';
  cout << ppoly3->area() << '\n';
  cin.get(); return 0;
}
```

```
20
10
0
```

- Shembulli 2:
  - Klasa që oërmban funksione virtuele, zakonisht është e njohur si **klasë polimorfike**

Klasat bazë abstrakte

- Klasat bazë abstrakte janë shumë të ngjashme me klasën **Polygon** të shembullit paraprak
  - Ata janë klasa që mund të shfrytëzohen vetëm si klasa bazë, andaj atyre I'u lejohet që të kenë funksione virtuele të padefinuara (*pure virtual functions*)
  - Sintaksa përcaktohet përmes shprehjes vijuese **=0**
- Klasa abstrakte Polygon do të duket si në vijim:

```
// abstract class CPolygon
class Polygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b; }
    virtual int area () =0;
};
```

Klasat bazë abstrakte

- Funksioni **area** është i padefinuar; andaj përmban simbolin **=0**, që e bën të jetë i tipit të ashtuquajtur **pure virtual**. Klasat që përmbajnë së paku një funksion të këtillë quhen **klasa bazë abstrakte**

- Tipi vijues i klasës abstrakte Polygon nuk mund të shfrytëzohet për të deklaruar objekte si:

```
Polygon mypolygon;   // not working if Polygon is abstract base class
```

- Por, por mund të shfrytëzohet për krijimin e pointerëve, dhe për shfrytëzimin e benificioneve nga vetia e polimorfizmit, si p.sh.:

```
Polygon * ppoly1;
Polygon * ppoly2;
```

Pointerët në klasën bazë

- Shembulli 3:

```
// pure virtual members can be called
// from the abstract base class
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b; }
    virtual int area() =0;
    void printarea()
      { cout << this->area() << '\n'; }
};

class Rectangle: public Polygon {
  public:
    int area (void)
      { return (width * height); }
};

class Triangle: public Polygon {
  public:
    int area (void)
      { return (width * height / 2); }
};
```

```
int main () {
  Rectangle rect;
  Triangle trgl;
  Polygon * ppoly1 = &rect;
  Polygon * ppoly2 = &trgl;
  ppoly1->set_values (4,5);
  ppoly2->set_values (4,5);
  ppoly1->printarea();
  ppoly2->printarea();
  cin.get(); return 0;
}
```

```
20
10
```

Klasat bazë abstrakte

- Shembulli 4:

```
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      { width=a; height=b; }
    virtual int area (void) =0;
};

class Rectangle: public Polygon {
  public:
    int area (void)
      { return (width * height); }
};

class Triangle: public Polygon {
  public:
    int area (void)
      { return (width * height / 2); }
};
```

```
int main () {
  Rectangle rect;
  Triangle trgl;
  Polygon * ppoly1 = &rect;
  Polygon * ppoly2 = &trgl;
  ppoly1->set_values (4,5);
  ppoly2->set_values (4,5);
  cout << ppoly1->area() << '\n';
  cout << ppoly2->area() << '\n';
  cin.get(); return 0;
}
```

```
20
10
```

Klasat bazë abstrakte

- Anëtarët virtuel dhe klasat abstrakte janë karakteristika të polimorfizmit në C++ dhe paraqesin avansimet e POO

- Në këtë rast vlenë të theksohet, shembujt paraprak sqarojnë vetëm mënyrën e implementimit, kurse shfrytëzimi real i tyre është i ndërlidhur me shfrytëzimin dinamik të memorjes

- Shembulli vijues, ilustron një rast të tillë!

Pointerët në klasën bazë

- Shembulli 5:

```
// dynamic allocation and polymorphism
#include <iostream>
using namespace std;

class Polygon {
  protected:
    int width, height;
  public:
    Polygon (int a, int b) : width(a), height(b) {}
    virtual int area (void) =0;
    void printarea()
      { cout << this->area() << '\n'; }
};

class Rectangle: public Polygon {
  public:
    Rectangle(int a,int b) : Polygon(a,b) {}
    int area()
      { return width*height; }
};

class Triangle: public Polygon {
  public:
    Triangle(int a,int b) : Polygon(a,b) {}
    int area()
      { return width*height/2; }
};
```

```
int main () {
  Polygon * ppoly1 = new Rectangle (4,5);
  Polygon * ppoly2 = new Triangle (4,5);
  ppoly1->printarea();
  ppoly2->printarea();
  delete ppoly1;
  delete ppoly2;
  cin.get(); return 0;
}
```

```
20
10
```

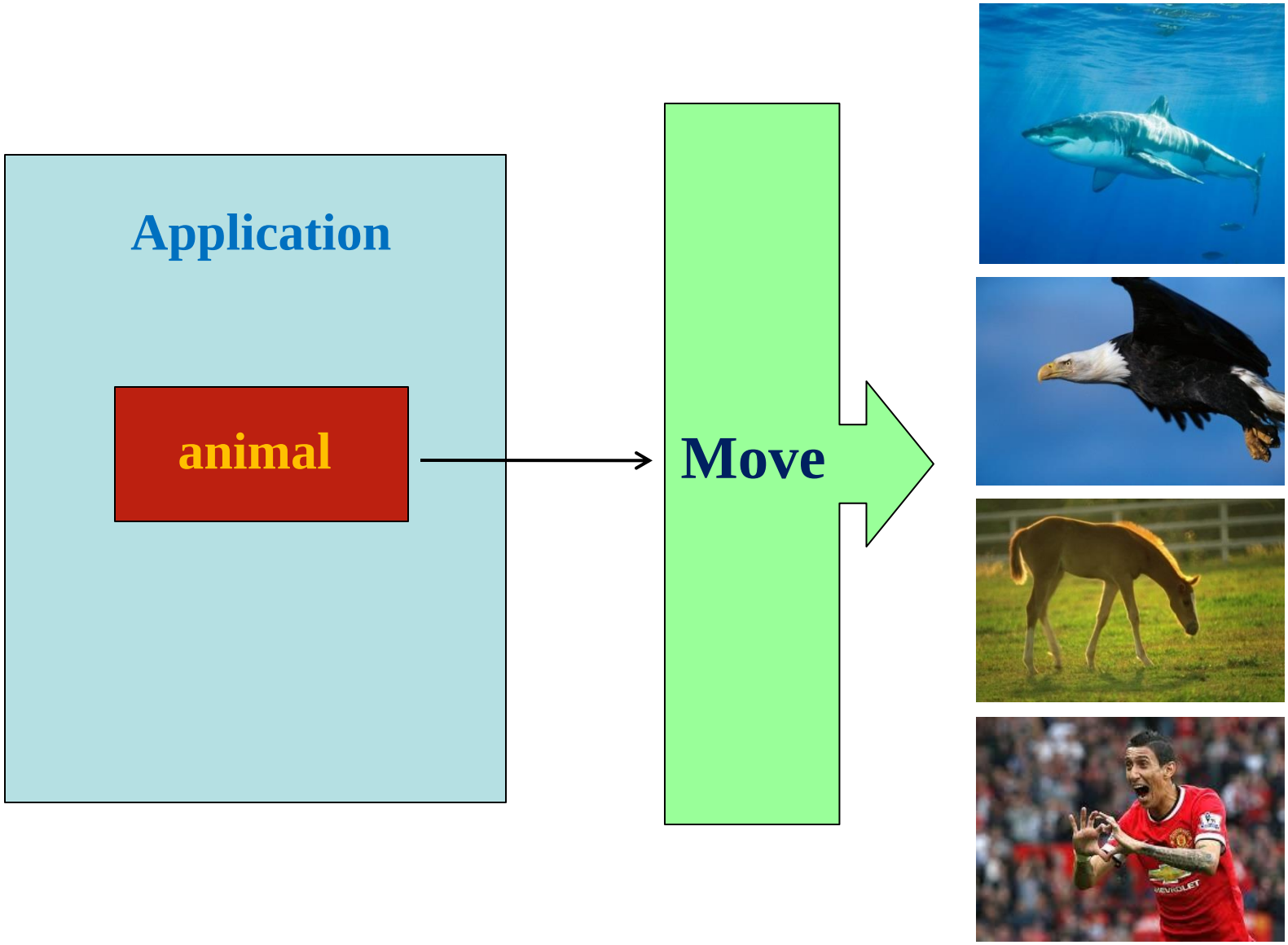
Polimorfizmi

- Programimi i njohur si "**Generic programming**" i referohet shfrytëzimit të kodit të njejtë për përfitim të rezultateve të ndryshme
  - *Shembull: Algoritmet për kërkim dhe sortim në tipe të ndryshme të të dhënave*

Polimorfizmi

- Në shembullin paraprak, lista e kafshëve në kuadër të lojës, mund të zgjerohet edhe me klasë të reja dhe të ketë të implementuar funksionin e lëvizjes

Polimorfizmi



- Pyetje eventuale?!

mentor.hamiti@universitetiaab.com