OOP: Polymorphism in C++

Part 1

Contents

- Concepts of Polymorphism
- Polymorphic Class and Virtual Functions

Concept of Polymorphism

a + b

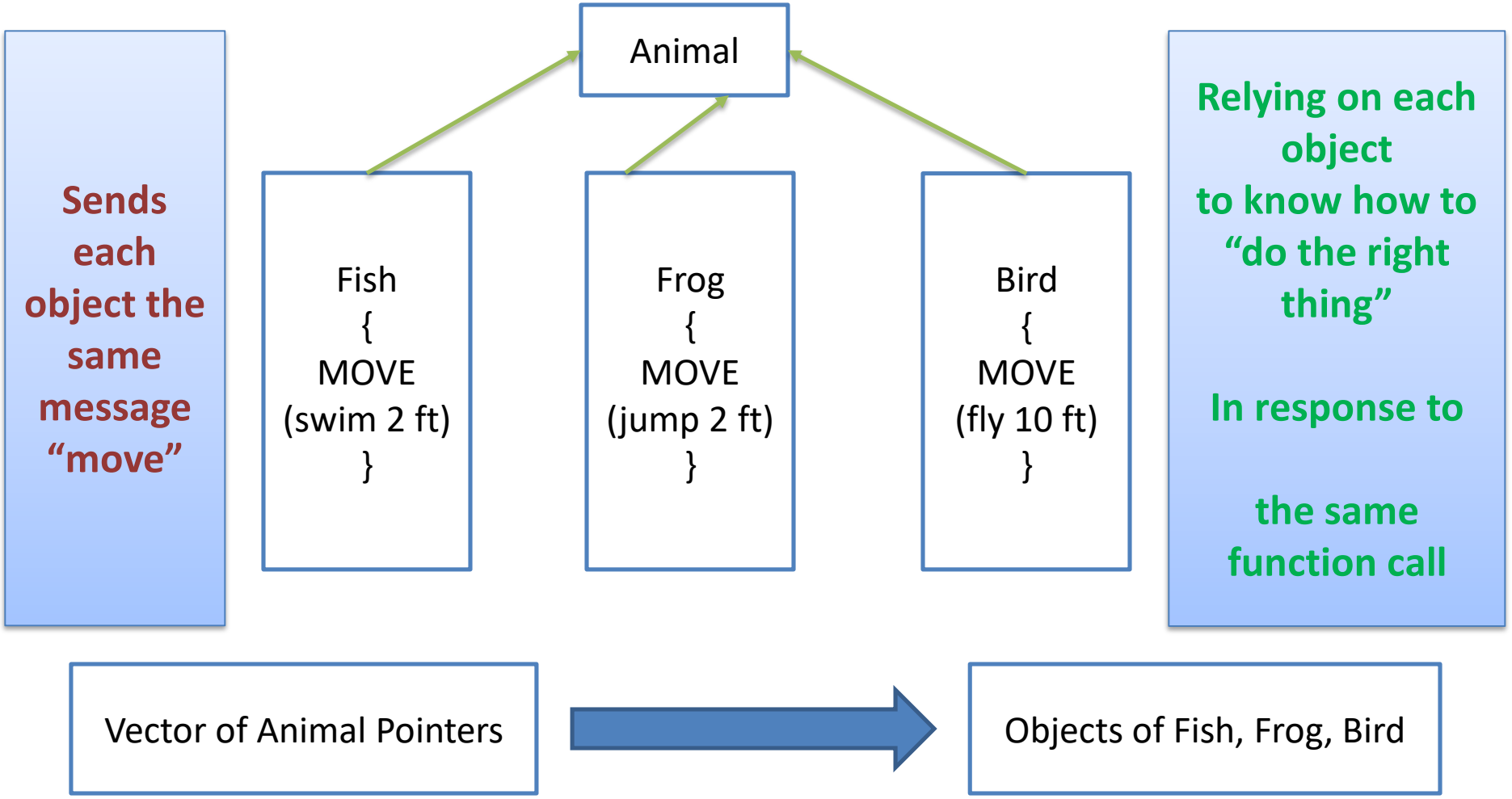
a=5
b=7 ➔ 12

a="Hello"
b="World"
➔ "HelloWorld"

- **Program in the general** rather than **program in the specific**

- **Objects** of classes that are part of the **same class hierarchy** as if they were all objects of the **hierarchy's base class**.

Concept of Polymorphism



Polymorphism: the same message sent to a variety of objects has "many forms" of results

Concept of Polymorphism

Polymorphism: Method Overloading

```
Class A
{
        show();
        show(int a);
}


Main()
{
        A obj;
        obj.show();
        obj.show(5);
}
```

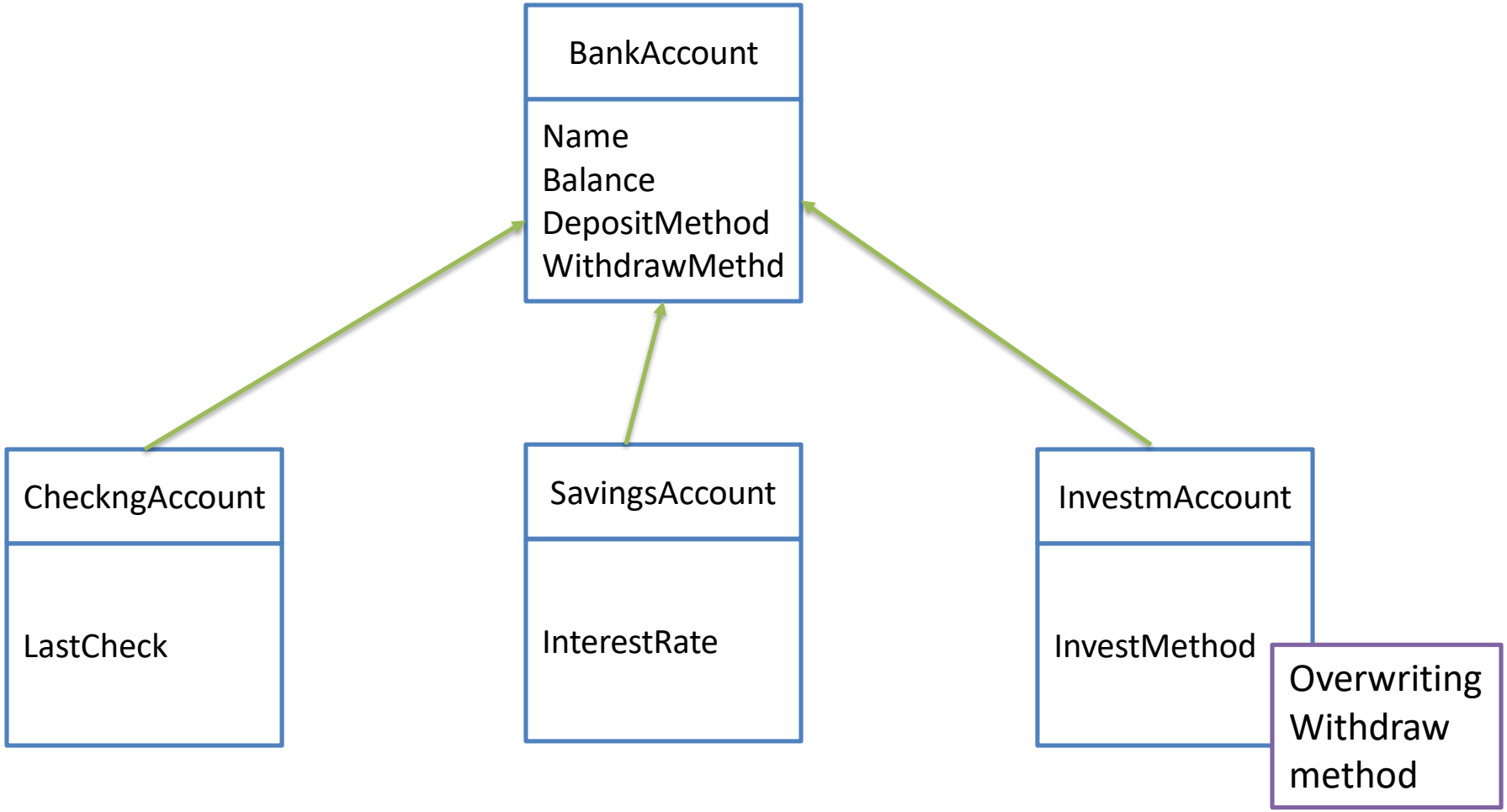
```
Class A
{
        show();
}

Class B: public A
{
        ??
}


Main()
{
        B obj;
        obj.show();
}
```

Polymorphism: Method Overwriting

Concept of Polymorphism



Polymorphism Advantages

- We can design and implement systems that are easily extensible
- **New classes** can be added with **little or no modification** to the general portions of the program, as long as the new classes are part of the inheritance hierarchy that the program processes generically.
- **Changes only** the parts of a program those **require direct knowledge** of the new classes
- Flexibility (right things at right time)

How: Get polymorphic behavior from base-class pointers aimed at derived-class objects

```cpp
#include <iostream>

using namespace std;

class Enemy
{
    protected:
        int dangerPower;

    public:
        void setDangerPower(int a)
        {
            dangerPower=a;
        }
};

class Samurai: public Enemy
{
    public:
        void hit()
        {
            cout << "This is samurai -" << dangerPower<<endl;
        }
};


class Alien: public Enemy
{
    public:
        void hit()
        {
            cout << "This is alien!!! -" << dangerPower<<endl;
        }

};
```

```cpp
int main()
{
    Samurai s;
    Alien a;

    Enemy *enemy1=&s;
    Enemy *enemy2=&a;

    enemy1->setDangerPower(25);
    enemy2->setDangerPower(100);

    s.hit();
    a.hit();
}
```

```cpp
#include <iostream>

using namespace std;

class Enemy
{
    public:
        virtual void hit()
        {

        }
};


class Samurai: public Enemy
{
    public:
        void hit()
        {
            cout<<"This is Samurai" << endl;
        }
};


class Alien: public Enemy
{
        void hit()
        {
            cout<<"This is Alien!!" << endl;
        }
};
```

```cpp
int main()
{
    Samurai s;
    Alien a;

    Enemy *enemy1=&s;
    Enemy *enemy2=&a;

    enemy1->hit();
    enemy2->hit();
}
```

Polymorphic Class

```cpp
#include <iostream>

using namespace std;

class Enemy
{
    public:
        virtual void hit()
        {
            cout<< "This is Enemy" << endl;
        }
};

class Samurai: public Enemy
{
    public:
        void hit()
        {
            cout<<"This is Samurai" << endl;
        }
};


class Alien: public Enemy
{

};
```

```cpp
int main()
{
    Samurai s;
    Alien a;

    Enemy *enemy1=&s;
    Enemy *enemy2=&a;

    enemy1->hit();
    enemy2->hit();
}
```

Pure Virtual Functions

```cpp
#include <iostream>

using namespace std;

class Enemy
{
    public:
        virtual void hit()=0;
};

class Samurai: public Enemy
{
    public:
        void hit()
        {
            cout<<"This is Samurai" << endl;
        }
};


class Alien: public Enemy
{
    public:
        void hit()
        {
            cout<<"This is Alien" << endl;
        }

};
```

Abstract Class

```cpp
int main()
{
    Samurai s;
    Alien a;

    Enemy *enemy1=&s;
    Enemy *enemy2=&a;

    enemy1->hit();
    enemy2->hit();
}
```

Pure Virtual Functions

Inheritance Exercise 1

```
class PPP
{
    int H:
  protected :
    int S;
  public :
    void input (int);
    void out();
};

class QQQ : private PPP
{
    int T;
  protected :
    int U;
  public :
    void indata(int, int);
    void outdata();
};

class RRR : public QQQ
{
    int M;
  public :
    void disp();
};
```

- Name the base class and derived class of the class **QQQ**.
- Name the data member(s) that can be accessed from function **disp()**.
- Name the member function(s), which can be accessed from the objects of class **RRR**.
- Is the member function **out()** accessible by the object of the class **QQQ?**

Inheritance Exercise 2

```
class Publisher
{
    char pub[12];
    double turnover;
  protected:
    void register();
  public:
    Publisher();
    void enter();
    void display();
};

class Branch
{
    char city[20];
  protected:
    float employees;
  public:
    Branch();
    void haveit();
    void giveit();
};

class Author : private Branch, public Publisher
{
    int acode;
    char aname[20];
    float amount;
  public:
    Author();
    void start();
    void show();
};
```

- Write the names of data members, which are accessible from objects belonging to class **Author**.
- Write the names of all the member functions which are accessible from objects belonging to class **Branch**.
- Write the names of all the members which are accessible from member functions of class **Author**.
- How many bytes will be required by an object belonging to class **Author**?

Inheritance Exercise 3

```
class Vehicle
{
  private:
    int wheels;
  protected :
    int passenger:
  public :
    void inputdata(int, int);
    void outputdata();
};

class Heavyvehicle : protected Vehicle
{
    int diesel_petrol;
  protected :
    int load;
  public:
    void readdata(int, int);
    void writedata();
};

class Bus : private Heavyvehicle
{
    char make[20];
  public :
    void fetchdata(char);
    void displaydata();
};
```

- Name the base class and derived class of the class **Heavyvehicle**.
- Name the data member(s) that can be accessed from function **displaydata()**.
- Name the data member's that can be accessed by an object of **Bus** class.
- Is the member function **outputdata()** accessible to the objects of **Heavyvehicle** class.

Inheritance Exercise 4

```
class Drug
{
    char category[10];
    char date_of_manufacture[10];
    char company[20];
  public:
    Drug();
    void enterdrugdetails();
    void showdrugdetails();
};

class Tablet : public Drug
{
  protected:
    char tablet_name[30];
    char volume_label[20];
  public:
    float price;
    Tablet();
    void entertabletdetails();
    void showtabletdetails ();
};

class PainReliever : public Tablet
{
    int dosage_units;
    char side_effects[20];
    int use_within_days;
  public:
    PainReliever();
    void enterdetails();
    void showdetails();
};
```

- How many bytes will be required by an object of class **Drug** and an object of class **PainReliever** respectively ?
- Write names of all the data members which are accessible from the object of class **PainReliever**.
- Write names of all the members accessible from member functions of class **Tablet**.
- Write names of all the member functions which are accessible from objects of class **PainReliever**.

Can you do It!

- Use your imagination and create a program in C++ which illustrates inheritance, multiple inheritance.

- Example: consider a family: the parents have two children, a girl and a boy, each of them inherits something different from mother, father, grandfather, grandmother, cousins, like the eyes color, the hair color….).