Chapter 9

More on Classes and Objects

Chapter 9 Topics (Part 1)

- Constructors（构造函数）
  - Constructor Declaration
  - Some Special Constructors
- Destructors（析构函数）
  - Destructor Declaration
  - Default Destructor
- Sequence of Invoking
- Array of Objects

Constructor Declaration

A constructor is a method that is used to initialize a newly constructed object.

SYNTAX

```
class ClassName {
public:
      ClassName (Parameter List)
    {
          ...
     }
          ...
};
```

Characteristics of Constructor

- has the same name as the class
- does not declare a return type
- is invoked（调用） automatically whenever memory space for an object is allocated

Constructor PlayingCard

```
class PlayingCard {
public:
        // constructor, initialize new playing card
        PlayingCard (Suits is, int ir)
        {  suitValue = is;
           rankValue = ir;
        }
        …
private:
        Suits suitValue;
        int rankValue;
};
```

Initializer

Constructors in C++ can use initializer (初始化器) to specify the initial values for dada members.

```
class PlayingCard {
public:
    PlayingCard (Suits is, int ir) : suitValue (is), rankValue (ir)
    {   }
    …
private:
    Suits suitValue;
    int rankValue;
};
```

A Default Constructor

is simply a constructor that takes no arguments.

If there is no user-defined constructor, the compiler will generate a default constructor.

```
ClassName::ClassName ( )
{ }
```

Overloaded Constructors

They are differentiated by the type signature:

```
class PlayingCard {
 public:
        PlayingCard ( ) // default constructor
        { suitValue = Diamond; rankValue = 1;}
        PlayingCard (Suits is, int ir) // constructor with two parameters
        { suitValue = is; rankValue = ir;}
       …
};
```

```
PlayingCard cardOne;//invoke default constructor
PlayingCard cardThree(PlayingCard::Club,6);//invoke constructor with two parameters
```

Constructor with Default Parameters

```
class PlayingCard {
 public:
       PlayingCard (Suits is, int ir=1) // constructor with one default parameter
       { suitValue = is; rankValue = ir;}
       …
};
```

```
PlayingCard cardTwo(PlayingCard::Heart); //use the default value
PlayingCard cardThree(PlayingCard::Club,6);
```

Ambiguous Call

```
class PlayingCard {
 public:
       PlayingCard (Suits is) // constructor with one parameter
       { suitValue = is; rankValue = 1;}
       PlayingCard (Suits is, int ir=1) // constructor with two parameters
       { suitValue = is; rankValue = ir;}
       …
};
```

```
PlayingCard cardTwo(PlayingCard::Heart);
//ambiguous call to overloaded functions
```

Chapter 9 Topics (Part 1)

- Constructors
  - Constructor Declaration
  - Some Special Constructors
- Destructors
  - Destructor Declaration
  - Default Destructor
- Sequence of Invoking
- Array of Objects

Destructor Declaration

A destructor is invoked when an object is deleted.

SYNTAX

```
class ClassName {
public:
      ~ ClassName (   )
     {
          …
       }
          …
};
```

Characteristics of Destructor

- is written as the name of the class preceded by a title (~)
- does not declare a return type
- does not take any arguments
- can not be overloaded
- is invoked automatically whenever memory space for an object is released

Destructor PlayingCard

```
class PlayingCard {
public:
        // destructor
        ~ PlayingCard ( )
        {
            …
        }
            …
};
```

A Default Destructor

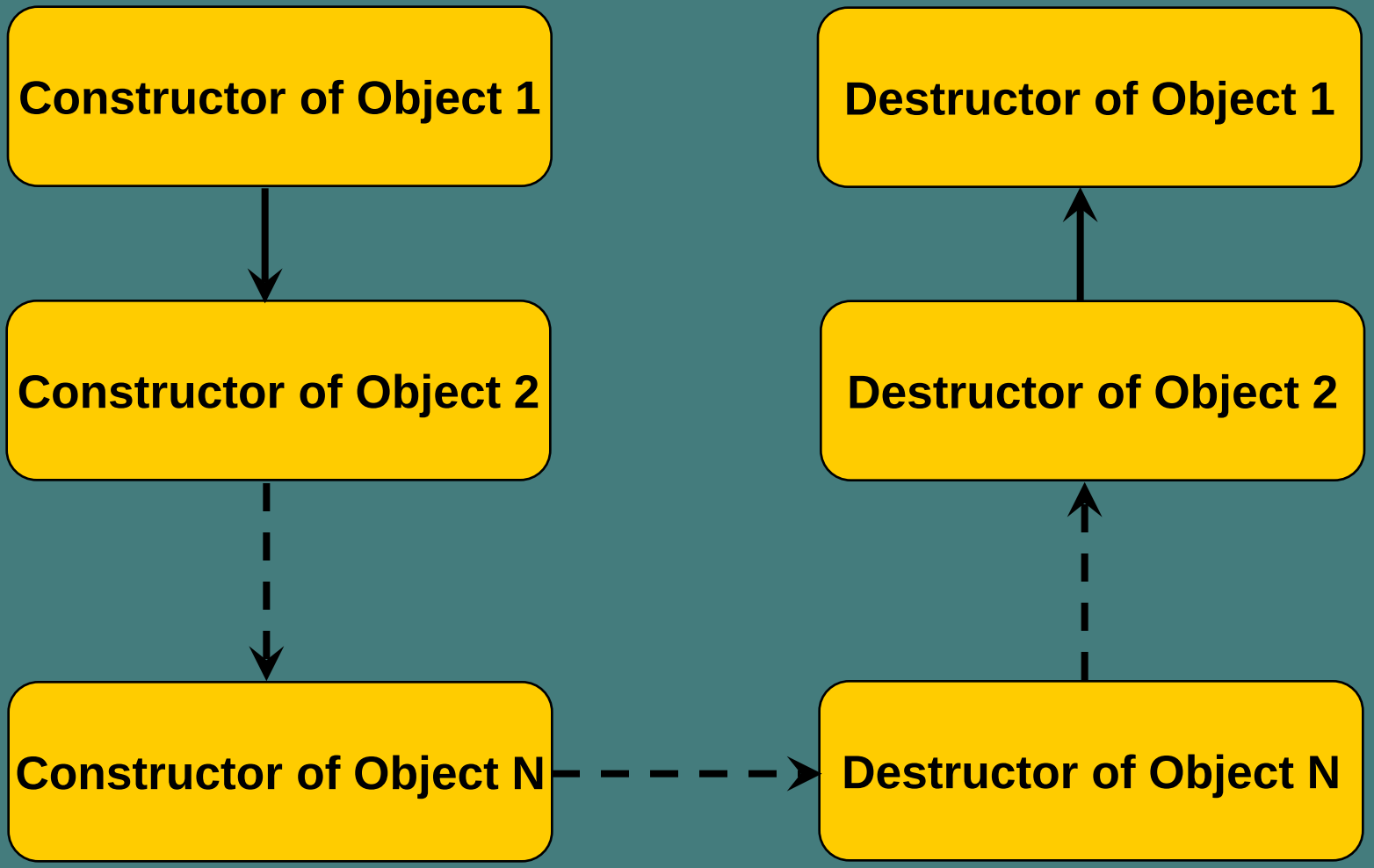
If there is no user-defined destructor, the compiler will generate a default destructor.

```
ClassName:: ~ ClassName (   )
{   }
```

Chapter 9 Topics (Part 1)

- Constructors
  - Constructor Declaration
  - Some Special Constructors
- Destructors
  - Destructor Declaration
  - Default Destructor
- Sequence of Invoking
- Array of Objects

Sequence of Invoking Constructors and Destructors



Tracing the Flow of Execution

```
void procedureA()
{
        Trace dummy("procedure A");
        procedureB(7);
}
void procedureB(int x)
{
        Trace dummy("procedure B");
        if (x<5){
                Trace aaa("true case in Procedure B");
        ...
        }
        else{
                Trace bbb("false case in Procedure B");
        ...
        }
}
```

```
class Trace{
public:
        //constructor
        Trace(string t):text(t)
        {cout<<"entering "<<text<<endl;}
        //destructor
        ~Trace()
        {cout<<"exiting "<<text<<endl;}
private:
        string text;
};
```

```
entering procedure A
entering procedure B
entering false case in Procedure B
  ...
exiting false case in Procedure B
exiting procedure B
exiting procedure A
```

Chapter 9 Topics (Part 1)

- Constructors
  - Constructor Declaration
  - Some Special Constructors
- Destructors
  - Destructor Declaration
  - Default Destructor
- Sequence of Invoking
- Array of Objects

Array of Objects

The creation of an array of objects presents:

- the allocation of the array itself
- the allocation of objects that the array will hold

SYNTAX

```
ClassName ArrayName[ConstIntExpression];
```

Initialize Array of Objects

- using the default constructor

```
PlayingCard cardArray[52];
```

- using constructor with parameters

```
PlayingCard cardArray[2]={
    PlayingCard(PlayingCard::Heart,1),
    PlayingCard(PlayingCard::Club,2)
};
```