Object Oriented Programming (OOP)
Lecture No. 10

Engr. Asma Khan
Assistant Professor
Software Engineering Dept.
NED University Karachi

Review

- Copy constructors
- Destructor
- Accessor Functions
- this Pointer

this Pointer

- There are situations where designer wants to return reference to current object from a function
- In such cases reference is taken from this pointer like (*this)

Example

```
Student Student::setRollNo(int aNo)
{
   …
   return *this;
}
Student Student::setName(char *aName)
{
   …
   return *this;
}
```

Example

```
int main()
{
    Student aStudent;
    Student bStudent;

    bStudent = aStudent.setName("Ahmad");
    …
    bStudent = aStudent.setName("Ali").setRollNo(2);

    return 0;
}
```

Separation of interface and implementation

- Public member function exposed by a class is called interface
- Separation of implementation from the interface is good software engineering

Complex Number

- There are two representations of complex number
  - Euler form
    - $z = x + i\,y$
  - Phasor form
    - $z = |z| (\cos\theta + i \sin\theta)$
    - z is known as the complex modulus and θ is known as the complex argument or phase

Example

Old implementation

Complex
float x float y
float getX() float getY() void setNumber (float i, float j) ...

New implementation

Complex
float z float theta
float getX() float getY() void setNumber (float i, float j) ...

Example

```
class Complex{ //old
   float x;
   float y;
public:
   void setNumber(float i, float j){
      x = i;
      y = j;
   }
   …
};
```

Example

```
class Complex{ //new
   float z;
   float theta;
public:
   void setNumber(float i, float j){
      theta = arctan(j/i);
      …
   }
   …
};
```

Advantages

- User is only concerned about ways of accessing data (interface)
- User has no concern about the internal representation and implementation of the class

Separation of interface and implementation

- Usually functions are defined in implementation files (.cpp) while the class definition is given in header file (.h)
- Some authors also consider this as separation of interface and implementation

Student.h

```
class Student{
   int rollNo;
public:
   void setRollNo(int aRollNo);
   int getRollNo();
   …
};
```

Student.cpp

```
#include “student.h”

void Student::setRollNo(int aNo){
   …
}
int Student::getRollNo(){
…
}
```

Driver.cpp

```
#include “student.h”

int main(){
   Student aStudent;
}
```

`const` Member Functions

- There are functions that are meant to be read only
- There must exist a mechanism to detect error if such functions accidentally change the data member

`const` Member Functions

- Keyword `const` is placed at the end of the parameter list

`const` Member Functions

Declaration:

```
class ClassName{
   ReturnVal Function() const;
};
```

Definition:

```
ReturnVal ClassName::Function() const{
   …
}
```

Example

```
class Student{
public:
   int getRollNo() const  {
      return rollNo;
   }
};
```

`const` Functions

- Constant member functions cannot modify the state of any object
- They are just *"read-only"*
- Errors due to typing are also caught at compile time

Example

```
bool Student::isRollNo(int aNo){
   if(rollNo == aNo){
      return true;
   }
   return false;
}
```

Example

```
bool Student::isRollNo(int aNo){
   /*undetected typing mistake*/
   if(rollNo = aNo){
      return true;
   }
   return false;
}
```

Example

```
bool Student::isRollNo
            (int aNo)const{
  /*compiler error*/
  if(rollNo = aNo){
    return true;
  }
  return false;
}
```

`const` Functions

- Constructors and Destructors cannot be `const`
- Constructor and destructor are used to modify the object to a well defined state

Example

```
class Time{
public:
  Time() const {}     //error…
  ~Time() const {}    //error…
};
```

`const` Function

- Constant member function cannot change data member
- Constant member function cannot access non-constant member functions

Example

```
class Student{
   char * name;
public:
   char *getName();
   void setName(char * aName);
   int ConstFunc() const{
      name = getName();//error
      setName("Ahmad");//error
   }
};
```

this Pointer and const Member Function

- this pointer is passed as constant pointer to const data in case of constant member functions

```
const Student *const this;
instead of
Student * const this;
```