

chapter 9 oop.

→ procedural programming :

→ uses variables to store data, and focuses on the processes / functions that occur in a program

→ Data and functions are separate and distinct

→ object-oriented-programming :

→ Based on objects that encapsulate the data and the functions that operate with and on the data.

آلية عمل المثلّات في هذا النوع من البرمجة /
يتم تقسيم المثلّة إلى عناصر ومن ثم إيجاد رابط يربط هذه العناصر
وليس تقسيم المثلّة إلى وظائف أصغر، وكلّ مثلّة على حدة تمّ تجميع
هذه المثلّات كائنات النوع السامع من البرمجة

• An object represents an entity in the real world that can be distinctly identified

مثال

مثل الطالب، المكتبة، الدائم، الزم

• An object has a unique identity, state, and behaviors.

حقوق البيانات

→ The state consists of a set of data fields (also known as properties) with their current values.

الخصائص

→ The behavior (also known as actions) of an object is defined by a set of methods.

ما يعني لكائن أن يفعله من وظائف وأشغال - يعرف السلوك داخل الكائن
البرمجي من خلال مجموعة من الدوال

circle class

class Name: circle

Data Fields:

radius is _____

Functions:

get Area

لأنه لا يمكن إنشاء
object من
class
لذلك
نحتاج
لـ object

circle object 1

Data Fields:

radius is 1.0

circle object 2

Data Fields:

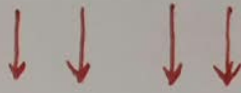
radius is 25

← two
objects of
the circle class.

طريقة الـ class الأساسية

- A class uses variables to define data fields and functions to define behaviors for objects.
 - A special type of functions, known as constructors are invoked to construct objects from the class.
- نوع خاص من الدوال يتم استدعائها لإنشاء كائن من الـ class

هذه هي الخطوات الأساسية من
تعريف أي class.



class circle → تعريف class (الاسم + النوع) (1)

{

public: → key word.

double radius; → Data field. (2)

circle ()

{

radius = 1;

}

circle (double newRadius)

{

radius = newRadius;

}

(3) Constructors (دالة)

• نفس اسم ال class.

• لا يتوي return type (لا يكتب قبل).

الاسم void أو ما تحاه ذلك.

• يتم استدعاها تلقائياً فور إنشاء

كائن جديد من ال class.

double getArea ()

{

return radius * radius * 3.14159 ;

}

};

(4)

→ Functions

• لاحظ في الكود السابق يوجد نوعين من Constructors :

Parameterized constructor

circle (double newRadius)



يسمح له بتعريف قيم مخصصة عند إنشاء الكائن
لنطبق القيم حسب رغبتك.

Default constructor

circle ()



ووظيفته إنشاء الكائن بغير افتراضية
تامة إذا لم يحددهم أي قيم
عند إنشاء الكائن.

في المثال السابق إفتراضية افتراضية
لنضع القطر ده ا.

معلومه برائتہ سے البرامیٹر لکھنا ضروری نہیں /
 البرامیٹر لائقہم بتعریفہ کلا فہم بتعریف المتغیرات الاخری ، فقط
 لکھنا نوعہ جانب امرہ داخل اُنو امرہ ال Function ..
 وھو یکتہ Variable Local (متغیر محلی) متغیر مؤقت وخصوصہ
 الوحیدہ ہی استقبال القیۃ القارۃ سے استہتم وقللوا اِلی المتغیر
 الاخری (فہی مثالاً radius) ، فہم یکتہ نوعہ مقاماً وفتنی سے
 الذاکہ منہ استہتم التفتہ

- A constructor with no parameters is referred to as a **no-arg constructor**.

- constructors execute automatically when a class object enters its scope.

لا یکتہ استہتم کبائی ال Functions لایہ یکتہ استہتم تلقائاً منہ
 لتعریف الکائنہ

طریقہ الاخری لتعریف Default Constructors /

class ↑ Circle :: Circle () : radius (i) { ↓ data field. }	constructor ↑ circle :: circle () { radius = 1; }
--	---

definition → التعريف
implementation → التنفيذ

◦ circle2 = circle1; → صيغة نسخة

* Separating Definition from Implementation %
(Class Abstraction)

[1]

class definition (.h)

- ↳ all the data fields.
- ↳ constructor prototypes.
- ↳ function prototypes.

[2]

class implementation (.cpp)

- ↳ constructors
- ↳ functions.

◦ Both files should have the same name, but with different extension names.

[3]

Test program. (driver program)

- ↳ Headers
- ↳ Test Functions.
- ↳ main.

Ex. Header file

* ifndef CIRCLE.H

* define CIRCLE.H

class circle

{

public:

circle();

circle(double);

double getArea();

double getRadius();

void setRadius(double);

private:

double radius;

};

* endif.

معلومات /

عند استخدام ملف في ملف آخر يجب

مراعاة ترتيب التعريف مرة واحدة فقط في

جميع الملفات (تُعرف داخل ملف واحد فقط)

Data Field Encapsulation:-

- أهمية هذا المفهوم في البرمجة الشيئية ؟!
- ١- التحكم في صفة البيانات (يمكننا منع تغيير قيم غير منطقية) .
 - ٢- سهولة الصيانة ومرونة (يمكننا تغيير طريقة عمل الدالة داخلياً دون أن نضطر لتغيير أي سطر كود في البرامج الخارجية التي تستخدم هذا الكلاس) .
 - ٣- إختصار التفاصيل الداخلية (المستعمل الخارجي لا يحتاج لمعرفة كيفية تخزين البيانات بل ماذا يمكنه فعله بها ، هنا يسهل استخدام الأنواع المعقدة) .

↳ Accessor and Mutator :-

A get function , getter (or accessor)

تسمح بقراءة قيم البيانات ، ترجع قيمة

A set function , setter (or mutator)

تسمح بتغيير قيمة البيانات .

Syntax

returnType get PropertyName ()

- if the returnType is bool , the get function should be defined as follows by convention :-

bool isPropertyName ()

- A set function has the following signature:

public void set propertyName (dataType property value)

↓
المعبر المراد لتعديل عليه

A simple circle class

Ex. objective: Demonstrate creating objects, accessing data and using functions.

```
* include <iostream>
using namespace std;
class circle
```

```
{
```

```
public:
```

```
circle circle ( )
```

```
{
```

```
double radius = 1;
```

```
}
```

```
circle ( double newRadius )
```

```
{
```

```
radius = new Radius ;
```

```
}
```

```
double get Area ( )
```

```
return radius * radius * 3.14159;
```

```
}
```

```
void set Radius ( double newRadius )
```

```
{
```

```
radius = new Radius ;
```

```
}
```

لا تستقبل ولا ترجع أي قيمة

وظيفة بناء وإزالة الكائن وإعطائ
قيمة ابتدائية افتراضية لتغير نصف القطر
عندما يتم استخدام كائن جديد

إعطائ قيمة لنصف القطر ليتم

بتدويرها المتجه

• تحويل لونا لتبني؟ لا يمكنه من

الحال قيمة لنصف القطر مباشرة عند تعريف

الكائن ومنه يسهل لتغير القيمة لاحقاً

لبريعة أخرى

• ماذا تستقبل؟

متغير يمثل لنصف القطر

• لا تقوم بإعطائ أي قيمة

وظيفة حساب مساحة الدائرة بناء على

قيمة لنصف القطر الخاصة بالكائن مباشرة

دونه الحاجة للتأجيل مقارنة لحساب يدوياً في كل

محتاج لحساب المساحة داخل main

لا تستقبل أي معاملات لأنها تستخدم لتغير خاصية الكائن مباشرة

تقوم بإعطائ قيمة عددية من نوع

double تمثل مساحة الدائرة

المسوية

objects

int main {

circle circle1 (1, 0);

circle circle2 (25);

circle circle3 (125);

نابع تنفيذ هذا البرنامج /

يتم إنشاء كائنات circle 1

يتم استدعاء الـ constructor الثاني

لأنه هنالك Argument وهو (1, 0)

تعتبر هذه القيمة في radius.

cout << "The area of the circle of radius" << circle1.radius <<

" is " << circle1.getArea() << endl;

نطبع 1, 0

circle2.radius = 100;

return 0;

}

استدعاء الدالة getArea

وبالتالي يتم طباعة

حجم مساحة الـ radius القيمة

object إلى عمل الاستدعاء

في النتيجة /

object3
↓
radius

object2
↓
radius

object1
↓
radius

Function

↑

↑

↑

شعار لجميع الكائنات

→ → inclusion guard.

```
#ifndef CIRCLE-H
```

```
#define CIRCLE-H
```

```
    {  
        أليس  
    }  
#endif
```

أعني استبعاد ملف في ملف آخر ، لتجنب
الاستبعاد أكثر من مرة .
هذه الطريقة في الاستبعاد :

• if n def ← إذا لم يتم استبعاد الملف بعد .

• define ← استبعاد الملف .

ملاحظة هامة : class definition تكتب هذه في ملف

→ → Inline Declaration.

When a function is implemented inside a class declaration (without
separate files) it automatically becomes an inline function.

This is also known as inline declaration.

ويعلم أيضاً ترميزه داخل implementation على الصيغة التالية /
File

```
inline double A::f2 ( )  
{  
    class A ← تم الالة  
}
```

المفهوم الجوهري بسيط :-

• `circle (double newRadius)` هذه الدالة تنقل مرة واحدة في عمر الكائن وهي لحظة إنشاء الكائن نفسه أول ما يولد في الذاكرة مثلاً عند كتابة `circle 2 (2.5)` : نقوم بإنشاء الكائن ووضع القيمة 2.5 لأول مرة.

ماذا لو امتلكنا بعد ذلك لغير القيمة (2.5) ؟
• `void setRadius (double newRadius)` هذه الدالة تنقل وتنقل بعد ما الكائن يكون استن وخلصه ويرجع لنا في كل مرة يكون محتاجة أكثر قيمة لضبط القيمة.

تُعطيني المرونة لتعديل خصائص الكائن في أي وقت يحتاجه بعد الانشاء سرعته الإصدار .
تنقل القيمة الجديدة
لا ترجع قيمة جديدة . فقط تنفذ عملية التعديل وتحت.

ex. 2 Tv is an object with

✓ state such as current channel ,
current volume , level and power
on or off

✓ behaviors such as change
channels , adjust volume and
turn on/off

TV
channel : int
volumeLevel : int
on : boolean
+ Tv ()
+ setChannel (newChannel : int)
+ setVolume (newVolume : int)
+ turnOn () : void
+ turnOff () : void
+ channelUp () : void
+ channel down () : void
+ volume up () : void
+ volume down () : void

Ex → Separating Definition from Implementation

الملف الأول

```
class circle
{
public:
    double radius;
    circle();
    circle(double);
    double getArea();
};
```

الملف الثاني

```
#include "circle.h"
circle::circle()
{ class pt Constructor الأول
```

```
    radius = 1;
}
circle::circle(double newRadius)
{ class pt Constructor الثاني
```

```
    radius = newRadius;
}
double circle::getArea()
{ class pt
```

```
    return radius * radius * 3.14159;
}
```

الملف الثالث

```
#include <iostream>
#include "circle.h"
using namespace std;
int main()
{
```

```
    circle circle1;
    circle circle2(5.0);
```

```
    cout << "Area of circle 1: " << circle1.getArea() << endl;
    cout << "Area of circle 2: " << circle2.getArea() << endl;
```

```
    return 0;
```

```
}
```

• class definition :

ifdef LOAN_H

define LOAN_H

class Loan

{

public:

Loan();

Loan(double rate, int years, double amount);

double getAnnualInterestRate();

int getNumberOfYears();

double getLoanAmount();

void setAnnualInterestRate(double rate);

void setNumberOfYears(int years);

void setLoanAmount(double amount);

double getMonthlyPayment();

double getTotalPayment();

private:

double annualInterestRate;

int numberOfYears;

double loanAmount;

};

endif

class Implementation in

include <iostream>

include <iomanip>

include "Loan.h"

using namespace std;

Loan::Loan() Constructor 1

{

annualInterestRate = 9.5;

numberOfYears = 30;

loanAmount = 100000;

}

Loan::Loan(double rate, int years, double amount) Constructor 2

{

annualInterestRate = rate;

numberOfYears = years;

loanAmount = amount;

}

double Loan::getAnnualInterestRate()

class . تم نوع المتغير الذي

القيمة

return annualInterestRate;

}

int Loan::getNumberOfYears()

{

return numberOfYears;

}

المعبر الموجود داخل class يكون رتي لكل ان class
العضو الشكر من موقعه وترسيم

```
double Loan::getLoanAmount()
```

```
{
```

```
    return loanAmount;
```

```
}
```

```
void Loan::setAnnualInterestRate(double rate)
```

```
{
```

```
    annualInterestRate = rate;
```

```
}
```

```
void Loan::setNumberOfYears(int years)
```

```
{
```

```
    numberOfYears = years;
```

```
}
```

```
void Loan::setLoanAmount(double amount)
```

```
{
```

```
    loanAmount = amount;
```

```
}
```

```
double Loan::getMonthlyPayment()
```

```
{
```

```
    double monthlyInterestRate = annualInterestRate / 1200;
```

```
    return loanAmount * monthlyInterestRate /
```

```
        (1 - (pow(1 / (1 + monthlyInterestRate), numberOfYears * 12)));
```

```
}
```

```
double Loan::getTotalPayment()
```

```
{
```

```
    return getMonthlyPayment() * numberOfYears * 12;
```

```
}
```

Test program :-

```
#include <iostream>
```

```
#include <iomanip>
```

```
#include "Loan.h"
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
cout << "Enter yearly interest rate : ";
```

```
double annualInterestRate;
```

```
cin >> annualInterestRate;
```

```
cout << "Enter number of years as an integer";
```

```
int numberofyears;
```

```
cin >> numberofyears;
```

```
cout << "Enter loan amount";
```

```
double loanAmount;
```

```
cin >> loanAmount;
```

عملية ائتمان

```
Loan loan (annualInterestRate, numberofyears, loanAmount);
```

class object القيمة المدخلة من قبل المستخدم

```
cout << fixed << setprecision(2);
```

```
cout << "The monthly payment is " << loan.getMonthlyPayment();
```

```
return 0;
```

دالة حساب قيمة الدفع الشهرية

```
}
```

Ex: The Loan class

Loan

- annualInterestRate: double
- numberOfYears: int
- loanAmount: double

المطابقان مع النوع كما متغير

+ Loan () Constructor 1

+ Loan (rate: double, years: int, amount: double) Constructor 2

+ get AnnualInterestRate (): double

+ get NumberOfYears (): int

+ get LoanAmount (): double

للمethod للبيانات
المعبرة في
private

+ set AnnualInterestRate (rate: double): void

+ set NumberOfYears (years: int): void

+ set LoanAmount (amount: double): void

+ get MonthlyPayment (): double

+ get TotalPayment (): double

البرهان

```
if (on && newVolumeLevel >= 1 && newVolumeLevel <= 7)
    volumeLevel = newVolumeLevel;
```

```
}
```

```
void channelUp()
```

```
{
```

```
    if (on && channel < 120)
```

```
        channel++;
```

```
}
```

```
void volumeUp()
```

تقرى العنقره

```
void volumeDown()
```

تقرى العنقره

```
}
```

```
int main()
```

```
{
```

```
Tv tv1; → إنشاء الكائن
```

```
tv1.turnOn();
```

```
tv1.setChannel(100);
```

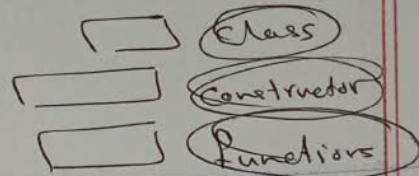
```
tv1.setVolume(3);
```

```
cout << "tv1's channel is" << tv1.channel << " and volume level is"
    << tv1.volumeLevel << endl;
```

```
return 0;
```

```
}
```

```
};
```



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

```
class TV
{
```

```
    TV( )
    {
```

```
        int channel = 1, volumeLevel = 1;
```

```
        bool on = 0;
```

```
    }
```

```
    void turn off ( ) →
```

```
    {
```

```
        on = false;
```

```
    }
```

```
    void turn on ( ) {
```

```
        on = true;
```

```
    }
```

```
    void set Channel (int new Channel) {
```

```
    {
```

```
        if (on && newChannel >= 1 && newChannel <= 120)
```

```
            Channel = newChannel;
```

```
    }
```

```
    void set Volume (int newVolume level)
```

```
    {
```

عادي جداً يكون عند دالة ما يستقبل
بالإختيار، وترك الأقول ما رقة ..
وعند الاستدعاء أيضاً تلعب الأوقاس
ما رقة ..

ضرباً في ٢٠