Object-oriented programming (OOP)
Lecture 1:Introduction to OOP

Dr. Abdullah Bokir

Object-oriented programming (OOP)

البرمجة الشيئية (OOP)

Lecture 1:Introduction to

المحاضرة الأولى: مقدمة إلى

OOP

عفوًا

Dr. Abdullah Bokir

صفحة (1)

Syllabus Overview

- Introduction
- Classes in OOP
- Attributes and Methods
- Objects in OOP
- Constructors
- Characteristics of OOP

Syllabus Overview

نظرة عامة على المنهج

Introduction

مقدمة

Classes in OOP

الفصول في OOP

Attributes and Methods

الصفات والأساليب

صفحة (2)

- Objects in OOP

- الكائنات في OOP

- Constructors

- البنائين

- Characteristics of OOP

- خصائص OOP

Syllabus Overview

- Access specifiers
- Exception Handling
- Events and Delegates
- Generics in C#

Syllabus Overview

نظرة عامة على المنهج

- Access specifiers

- محددات الوصول

- Exception Handling

- التعامل مع الاستثناءات

- Events and Delegates

الفعاليات والوفود

صفحة (3)

Generics in C#

الجينات في C#

Introduction to OOP

- First let’s understand the meaning of the term object-oriented programming.

- **object**. According to the dictionary, an object is something that can be seen, felt, or touched; something that has physical existence in the real world

Introduction to OOP

مقدمة إلى OOP

First let’s understand the meaning of the term object-oriented programming.

دعونا أولاً نفهم معنى مصطلح البرمجة الشيئية.

object. According to the dictionary, an object is something that can be seen, felt, or

هدف. وفقا للقاموس، الكائن هو شيء يمكن رؤيته، أو الشعور به، أو

touched; something that has physical

لمست. شيء له المادية

صفحة (4)

existence in the real world

الوجود في العالم الحقيقي

صفحة (4)

Introduction to OOP

- **Oriented**, which indicates a direction or something to aim for.

- **Programming** is just giving instructions to the computer to do a particular thing.

Introduction to OOP

مقدمة إلى OOP

Oriented, which indicates a direction or something to aim for.

موجه، وهو ما يشير إلى اتجاه أو شيء يهدف إليه.

Programming is just giving instructions to the computer to do a particular thing.

البرمجة هي مجرد إعطاء تعليمات للكمبيوتر للقيام بشيء معين.

Introduction to OOP

- The phrase **object-oriented programming. OOP** means that we write our computer programs by keeping objects at the center of our thinking.

- OOP is neither a tool nor a programming language—it is just a concept.

Introduction to OOP

مقدمة إلى OOP

The phrase object-oriented programming. OOP means that we write our computer programs by keeping objects at the center of our thinking.

عبارة البرمجة الشيئية. OOP يعني أننا نكتب برامج الكمبيوتر الخاصة بنا عن طريق إبقاء الأشياء في مركز تفكيرنا.

OOP is neither a tool nor a programming language—it is just a concept.

OOP ليست أداة ولا لغة برمجة، بل هي مجرد مفهوم.

Introduction to OOP

- C#, Java, C++, Python, … are examples of OOP languages.

- Other programming concepts in the programming world, such as **procedural programming** such as C language, **functional programming** such as F# language.

Introduction to OOP

مقدمة إلى OOP

C#, Java, C++, Python, … are examples of OOP languages.

C#، Java، C++، Python، ... هي أمثلة على لغات OOP.

Other programming concepts in the programming world, such as procedural programming such as C language, functional programming such as F# language.

مفاهيم برمجة أخرى في عالم البرمجة، مثل البرمجة الإجرائية مثل لغة C، والبرمجة الوظيفية مثل لغة F#.

Classes in OOP

- A class is like a template or blueprint that tells us what properties and behaviors an instance of this class will have.

- In most circumstances, a class itself can't actually do anything—it is just used to create objects.

Classes in OOP

دروس في OOP

A class is like a template or blueprint that tells us what properties and behaviors an instance of this class will have.

يشبه الفصل قالبًا أو مخططًا يخبرنا بالخصائص والسلوكيات التي سيحتوي عليها مثيل هذا الفصل.

In most circumstances, a class itself can't actually do anything—it is just used to create objects.

في معظم الظروف، لا تستطيع الفئة نفسها فعل أي شيء، فهي تستخدم فقط لإنشاء الكائنات.

Classes in OOP

- **Example**: Human class. Here, when we say Human, we don't mean any particular person, but we are referring to a human being in general. A human that has two hands, two legs, and a mouth, and which can also walk, talk, eat, and think. These properties and their behaviors are applicable to most human beings.

Classes in OOP

دروس في OOP

- Example: Human class. Here, when we say Human, we don't mean any particular person, but we are referring to a human being in general. A human that has two hands, two legs, and a mouth, and which can also walk, talk, eat, and think. These properties

- مثال: فئة الإنسان. وهنا عندما نقول "إنسان" فإننا لا نقصد أي شخص بعينه، بل نشير إلى الإنسان بشكل عام. إنسان له يدان ورجلان وفم، ويستطيع أيضًا المشي والكلام والأكل والتفكير. هذه الخصائص وسلوكياتها تنطبق على معظم البشر.

Classes in OOP

- There are hundreds of properties that you can list for a human, for example:
  - Height
  - Weight
  - Age

Classes in OOP

دروس في OOP

There are hundreds of properties that you can list for a human, for example:

هناك المئات من الخصائص التي يمكنك سردها للإنسان، على سبيل المثال:

- Height

- ارتفاع

- Weight

- وزن

صفحة (10)

- Age

- عمر

Classes in OOP

- Similarly there are hundreds of particular behaviors that a person can perform, for example
  - Walk
  - Talk
  - Eat

Classes in OOP

دروس في OOP

 Similarly there are hundreds of particular

 وبالمثل هناك المئات من خاصة

behaviors that a person can perform, for example ▪ Walk

السلوكيات التي يمكن أن يقوم بها الإنسان، على سبيل المثال ▪ المشي

▪ Talk

- يتحدث

صفحة (11)

- Eat

- يأكل

صفحة (11)

The general form of a class in C#

- To create a class in C# language you need to follow the following syntax:

```
class class-name {
    // this is class body
}
```

- The class phrase is a **reserved keyword** in C#, and it is used to tell the compiler that we want to create a class.

The general form of a class in C#

الشكل العام للفصل في لغة C#

To create a class in C# language you need to follow the following syntax:

لإنشاء فصل دراسي بلغة C#، عليك اتباع بناء الجملة التالي:

The class phrase is a reserved keyword in C#, and it is used to tell the compiler that we want to

عبارة الفئة هي كلمة رئيسية محجوزة في C#، ويتم استخدامها لإخبار المترجم أننا نريد ذلك

create a class.

إنشاء فصل دراسي.

The general form of a class in C#

- To create a class, place the **class** keyword and then the name of the class after a space.
- The name of the class can be anything that starts with a character or an underscore. We can also include numbers in the class name, but not the first character of a class name.
- After the chosen name of the class, you have to put an opening curly brace, which denotes the start of the class body.

The general form of a class in C#

الشكل العام للفصل في لغة C#

To create a class, place the class keyword and then the name of the class after a space.

لإنشاء فصل دراسي، ضع الكلمة الأساسية للفئة ثم اسم الفصل بعد مسافة.

The name of the class can be anything that starts with a character or an underscore.
We can also include numbers in the class name, but not the first character of a class

يمكن أن يكون اسم الفئة أي شيء يبدأ بحرف أو بشرطة سفلية. يمكننا أيضًا تضمين أرقام في اسم الفئة، ولكن ليس الحرف الأول من اسم الفئة.

After the chosen name of the class, you have to put an opening curly brace, which denotes the start of the class body.

بعد اسم الفصل المختار، عليك وضع قوس مفتوح متعرج، والذي يشير إلى بداية جسم الفصل.

The general form of a class in C#

- You can add content in the class, such as properties and methods
- then finish the class with a closing curly brace as follows.

```
class class-name {
  // property 1
  // property 2
  // ...

  // method 1
  // method 2
  // ...
}
```

The general form of a class in C#

الشكل العام للفصل في لغة C#

You can add content in

يمكنك إضافة محتوى في

the class, such as

الطبقة، مثل

properties and methods

صفحة (14)

then finish the class with a closing curly brace as

ثم قم بإنهاء الفصل بقوس إغلاق متعرج مثل

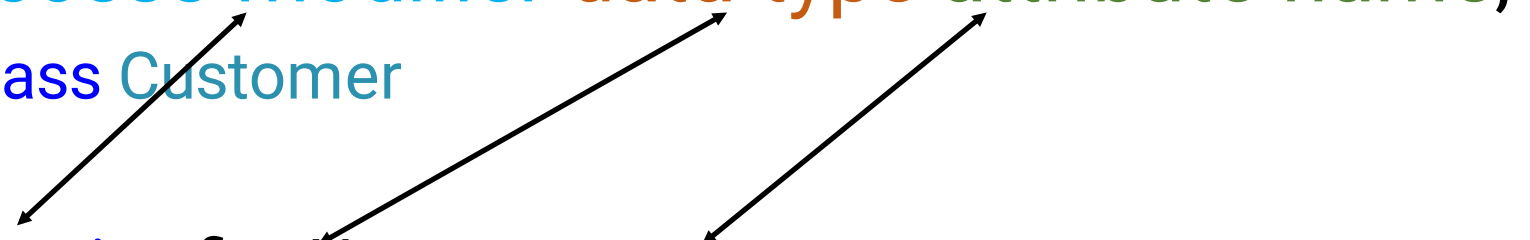
follows.

يتبع.

Attributes in a class

- Attributes are like variables and can be defined using the following syntax

access-modifier data-type attribute-name;



```
public class Customer
{
    public string firstName;
    public string lastName;
    public string phoneNumber;
    public string emailAddress;
}
```

Attributes in a class

السمات في الصف

 Attributes are like variables and can be defined using the following syntax

 السمات تشبه المتغيرات ويمكن تعريفها باستخدام الصيغة التالية

publicstring firstName;

الاسم الأول للسلسلة العامة؛

publicstring lastName;

اسم العائلة publicstring؛

صفحة (15)

publicstring phoneNumber;

رقم هاتف السلسلة العامة؛

publicstring emailAddress;

عنوان البريد الإلكتروني publicstring؛

Methods in a class

- A method is a piece of code that is written in the code file and can be reused.

- A method can hold many lines of code, which will be executed when it is called. Let's take a look at the general form of a method:

```
access-modifier return-type method-name(parameter-list) {
    // method body
}
```

Methods in a class

الأساليب في الصف

A method is a piece of code that is written in the code file and can be reused.

الطريقة هي جزء من التعليمات البرمجية المكتوبة في ملف التعليمات البرمجية ويمكن إعادة استخدامها.

A method can hold many lines of code, which will be executed when it is called. Let's take a look at the general form of a method:

يمكن أن تحتوي الطريقة على عدة أسطر من التعليمات البرمجية، والتي سيتم تنفيذها عند استدعائها. دعونا نلقي نظرة على الشكل العام للطريقة:

Methods in a class

- We can see that the first thing in the method declaration is an **access-modifier**. This will set the access permission of the method.
- Then, we have the **return-type** of the method, which will hold the type that the method will return, such as string, int, double, or another type.
- After that, we have the **method-name** and then brackets, **()**, which indicate that it is a method.

Methods in a class

الأساليب في الصف

We can see that the first thing in the method declaration is an access-modifier. This will set the access permission of the method.

يمكننا أن نرى أن أول شيء في إعلان الطريقة هو معدّل الوصول. سيؤدي هذا إلى تعيين إذن الوصول للطريقة.

Then, we have the return-type of the method, which will hold the type that the method will

بعد ذلك، لدينا نوع الإرجاع للطريقة، والذي سيحتوي على النوع الذي ستحتفظ به الطريقة

return, such as string, int, double, or another type.

return، مثل string أو int أو double أو أي نوع آخر.

After that, we have the method-name and then brackets, (), which indicate that it is a method.

بعد ذلك، لدينا اسم الطريقة ثم الأقواس () التي تشير إلى أنها طريقة.

Methods in a class

- In the brackets, we have the **parameter-list**. This can either be empty or can contain one or more parameters.
- Finally, we have curly brackets, **{}**, which hold the **method body**. The code that the method will execute goes inside here.

Methods in a class

الأساليب في الصف

In the brackets, we have the parameter-list. This can either be empty or can contain one or more parameters.

بين قوسين، لدينا قائمة المعلمات. يمكن أن يكون هذا فارغًا أو يمكن أن يحتوي على معلمة واحدة أو أكثر.

Finally, we have curly brackets, {}, which hold the method body. The code that the method will execute goes inside here.

أخيرًا، لدينا قوسان متعرجان، {}، يحملان نص الطريقة. الكود الذي ستنفذه الطريقة موجود هنا.

Creating a method

```
public string GetFullName()
{
    return firstName + " " + lastName;
}
```

Creating a method

إنشاء طريقة

publicstring GetFullName()

سلسلة عامة GetFullName()

return firstName + " " + lastName; }

إرجاع الاسم الأول + "" + الاسم الأخير؛ }

Writing a simple class

- Let's imagine that we are developing some software for a bank. Our application should keep track of the bank's customers and their bank accounts, and perform some basic actions on those bank accounts.

- As we are going to design our application using C#, we have to think of our application in an object- oriented way.

Writing a simple class

كتابة فئة بسيطة

Let's imagine that we are developing some

دعونا نتخيل أننا نقوم بتطوير بعض

software for a bank. Our application should keep track of the bank's customers and their bank accounts, and perform some basic actions on those bank accounts.

برنامج للبنك. يجب أن يقوم تطبيقنا بتتبع عملاء البنك وحساباتهم المصرفية، وتنفيذ بعض الإجراءات الأساسية على تلك الحسابات المصرفية.

As we are going to design our application using C#, we have to think of our application in an object- oriented way.

بما أننا سنقوم بتصميم تطبيقنا باستخدام لغة C#، علينا أن نفكر في تطبيقنا بطريقة موجهة للكائنات.

Writing a simple class

- Some objects that we will need for this application could be a customer object, a bank account object, and other objects.

- So, to make blueprints of these objects, we have to create a Customer class and a BankAccount class

Writing a simple class

كتابة فئة بسيطة

Some objects that we will need for this

بعض الأشياء التي سنحتاجها لهذا الغرض

application could be a customer object, a bank account object, and other objects.

يمكن أن يكون التطبيق كائن عميل، وكائن حساب مصرفي، وكائنات أخرى.

So, to make blueprints of these objects, we have to create a Customer class and a BankAccount class

لذلك، لعمل مخططات لهذه الكائنات، يتعين علينا إنشاء فئة العملاء وفئة BankAccount

Writing a simple class

```
class Customer
{
    public string firstName;
    public string lastName;
    public string phoneNumber;
    public string emailAddress;

    public string GetFullName()
    {
        return FirstName + " " + LastName;
    }
}
```

Writing a simple class

كتابة فئة بسيطة

publicstring firstName;

الاسم الأول للسلسلة العامة؛

publicstring lastName;

اسم العائلة publicstring؛

publicstring phoneNumber;

رقم هاتف السلسلة العامة؛

```
publicstring emailAddress;
```

```
عنوان البريد الإلكتروني publicstring؛
```

```
publicstring GetFullName()
```

```
سلسلة عامة GetFullName()
```

```
return FirstName + " " + LastName; }
```

```
إرجاع الاسم الأول + "" + اسم العائلة؛ }
```

Writing a simple class

- We started with the class keyword and then the name of the class, which is Customer. After that, we added the class body inside curly braces, {}.
- The variables that the class has are firstName, lastName, phoneNumber, and emailAddress.
- The class also has a method called GetFullName(), which uses the firstName and the lastName fields to prepare the full name and return it.

Writing a simple class

كتابة فئة بسيطة

We started with the class keyword and then the name of the class, which is Customer. After that, we added the class body inside curly braces, {}.

لقد بدأنا بالكلمة الأساسية للفئة ثم اسم الفئة، وهو العميل. بعد ذلك، أضفنا نص الفصل داخل الأقواس المتعرجة، {}.

The variables that the class has are firstName, lastName, phoneNumber, and emailAddress.

المتغيرات الموجودة في الفئة هي الاسم الأول، الاسم الأخير، رقم الهاتف، وعنوان البريد الإلكتروني.

The class also has a method called GetFullName(), which uses the firstName and the lastName fields to prepare the full name and return it.

لدى الفئة أيضًا طريقة تسمى GetFullName()، والتي تستخدم حقلي الاسم الأول واسم العائلة لتحضير الاسم الكامل وإعادته.

Writing a simple class

```
class BankAccount
{
   public string bankAccountNumber;
   public string bankAccountOwnerName;
   public double amount;
   public DateTime openningDate;

   public string Credit()
   {
      return "amount credited";
   }

   public string Debit()
   {
      return "amount debited";
   }
}
```

Writing a simple class

كتابة فئة بسيطة

publicdouble amount;

مبلغ مزدوج عام؛

public DateTime openningDate;

التاريخ والوقت العام openningDate؛

publicstring Credit()

رصيد السلسلة العامة ()

صفحة (24)

return"amount credited";

إرجاع"المبلغ المضاف";

publicstring Debit()

الخصم publicstring ()

return"amount debited";

إرجاع"المبلغ المدين";

Writing a simple class

- Here, the name of the class is BankAccount
- fields of the class are bankAccountNumber, bankAccountOwnerName, amount, and openningDate
- Two methods, Credit and Debit are defined in the class.

Writing a simple class

كتابة فئة بسيطة

Here, the name of the class is BankAccount

هنا، اسم الفئة هو BankAccount

fields of the class are bankAccountNumber,

حقول الفئة هي BankAccountNumber،

bankAccountOwnerName, amount, and

اسم البنك الحسابي والمبلغ و

صفحة (25)

openningDate

تاريخ الافتتاح

Two methods, Credit and Debit are defined in the class.

يتم تعريف طريقتين، الائتمان والخصم في الفصل.

Objects in OOP

- An object is an instance of a class. In other words, an object is an implementation of a class.

- For example, in our banking application, we have a Customer class, but that doesn't mean that we actually have a customer in our application

Objects in OOP

الكائنات في OOP

An object is an instance of a class. In other words, an object is an implementation of a class.

الكائن هو مثيل لفئة. بمعنى آخر، الكائن هو تطبيق لفئة ما.

For example, in our banking application, we have a Customer class, but that doesn't mean that we actually have a customer in our application

على سبيل المثال، في تطبيقنا المصرفي، لدينا فئة العملاء، لكن هذا لا يعني أن لدينا بالفعل عميلًا في تطبيقنا

Objects in OOP

- To create a customer, we have to create an object of the Customer class.

- Let's say that we have a customer called Mr. Jack Jones. For this customer, we have to create an object of the Customer class, where the name of the person is Jack Jones.

Objects in OOP

الكائنات في OOP

To create a customer, we have to create an object of the Customer class.

لإنشاء عميل، علينا إنشاء كائن من فئة العميل.

Let's say that we have a customer called Mr. Jack Jones. For this customer, we have to create an object of the Customer class, where the name of the person is Jack Jones.

لنفترض أن لدينا عميلًا يُدعى السيد جاك جونز. بالنسبة لهذا العميل، يتعين علينا إنشاء كائن من فئة العملاء، حيث يكون اسم الشخص هو جاك جونز.

How to create objects

- In C#, to create an object of a class, you have to use the **new** keyword. As shown in the following syntax

```
class-name object-name  = new class-name();
```

- Example

```
Customer jackJones = new Customer();
```

How to create objects

كيفية إنشاء الكائنات

In C#, to create an object of a class, you have to use the new keyword. As shown in the following syntax

في C#، لإنشاء كائن من فئة، عليك استخدام الكلمة الأساسية الجديدة. كما هو موضح في بناء الجملة التالي

class-name object-name = newclass-name();

اسم الفئة اسم الكائن = newclass-name();

Example

مثال

صفحة (28)

Assigning values to the variables of the object

```
public class Program
{
    static void Main(string[] args)
    {
        Customer customer1 = new Customer();
        customer1.firstName = "jack";
        customer1.lastName = "Jones";
        customer1.phoneNumber = "1234567890";
        customer1.emailAddress = "jackJones@gmail.com";
        Console.WriteLine("customer's full name is: "+ customer1.GetFullName());
    }
}
```

Assigning values to the variables of the

إسناد القيم للمتغيرات

object

هدف

Customer customer1 = new Customer();

العميل customer1 = العميل الجديد();

customer1.firstName = "jack";

```
customer1.firstName = "جاك";
```

```
customer1.lastName = "Jones";
```

```
customer1.lastName = "جونز";
```

```
Console.WriteLine("customer's full name is: "+ customer1.GetFullName()); }
```

```
Console.WriteLine("الاسم الكامل للعميل هو:"+ customer1.GetFullName()); }
```

Constructor of a class

- In every class, there is a special type of method, called a **constructor**.
- You can create a constructor in a class and program it [This is called **Explicit Constructor**]. If you don't create one yourself, the compiler will create a very simple constructor and use that instead [This is called **Implicit Constructor**].

Constructor of a class

منشئ فئة

In every class, there is a special type of method, called a constructor.

في كل فئة، هناك نوع خاص من الأساليب، يسمى المنشئ.

You can create a constructor in a class and program it [This is called Explicit Constructor]. If you don't create one yourself, the compiler will create a very simple

يمكنك إنشاء منشئ في فصل دراسي وبرمجته [وهذا ما يسمى منشئ صريح]. إذا لم تقم بإنشاء واحد بنفسك، فسيقوم المترجم بإنشاء مُنشئ بسيط للغاية ويستخدمه بدلاً من ذلك [يُسمى هذا المُنشئ الضمني].

Constructor of a class

- The name of the constructor should be the same as the class name.

- A constructor doesn't have a return type. This is because a constructor can't return anything; it's for initialization, not for any other type of action

Constructor of a class

منشئ فئة

The name of the constructor should be the same as the class name.

يجب أن يكون اسم المنشئ هو نفس اسم الفئة.

A constructor doesn't have a return type. This is because a constructor can't return anything; it's for initialization, not for any other type of action

لا يحتوي المُنشئ على نوع إرجاع. وذلك لأن المُنشئ لا يمكنه إرجاع أي شيء؛ إنه للتهيئة، وليس لأي نوع آخر من الإجراءات

Constructor of a class

- A constructor is a method that gets triggered when an object of a class is created.

- A constructor is mainly used to set the prerequisites of the class.

Constructor of a class

منشئ فئة

A constructor is a method that gets triggered when an object of a class is created.

المُنشئ هو أسلوب يتم تشغيله عند إنشاء كائن من فئة ما.

A constructor is mainly used to set the prerequisites of the class.

يتم استخدام المُنشئ بشكل أساسي لتعيين المتطلبات الأساسية للفئة.

Constructor of a class

- For example, if you are creating an object of the Human class, that human object must have a date of birth. You can set this requirement in the constructor.

- You can also configure the constructor to set the date of birth as today if no date of birth is given. This depends on the needs of your application.

Constructor of a class

منشئ فئة

For example, if you are creating an object of the Human class, that human object must have a date of birth. You can set this requirement in the constructor.

على سبيل المثال، إذا كنت تقوم بإنشاء كائن من فئة "البشر"، فيجب أن يكون لهذا الكائن البشري تاريخ ميلاد. يمكنك تعيين هذا المطلب في المنشئ.

You can also configure the constructor to set the date of birth as today if no date of birth is given. This depends on the needs of your application.

يمكنك أيضًا تكوين المُنشئ لتعيين تاريخ الميلاد كما هو اليوم إذا لم يتم تحديد تاريخ ميلاد. هذا يعتمد على احتياجات التطبيق الخاص بك.

Constructor of a class

- Let's take a look at the general form of a constructor, as shown in the following code:

```
access-modifier class-name(parameter-list) {
    // constructor body
}
```

Constructor of a class

منشئ فئة

 Let's take a look at the general form of a constructor, as shown in the following code:

 دعونا نلقي نظرة على الشكل العام للمنشئ، كما هو موضح في الكود التالي:

Example of a Constructor (parameterless Constructor)

```
class BankAccount
{
    public string owner;
    public BankAccount()
    {
        owner = "Some person";
    }
}
```

Example of a Constructor (parameterless Constructor)

مثال على المنشئ (منشئ بدون معلمات)

publicstring owner;

مالك السلسلة العامة؛

publicBankAccount()

حساب البنك العام ()

owner = "Some person";

```
المالك = "شخص ما";
```

Example of a Constructor (parameterized Constructor)

- We can also take the name of the owner of the bank

account as a parameter in the constructor and use it to assign the variable, as shown in the following code:

```
class BankAccount
{
    public string owner;
    public BankAccount(string theOwner)
    {
        owner = theOwner;
    }
```

Example of a Constructor (parameterized Constructor)

مثال على المنشئ (منشئ ذو معلمات)

 We can also take the name of the owner of the bank

 يمكننا أيضًا أخذ اسم صاحب البنك

account as a parameter in the constructor and use it to assign the variable, as shown in the following code:

account كمعامل في المُنشئ واستخدامه لتعيين المتغير، كما هو موضح في الكود التالي:

publicstring owner;

مالك السلسلة العامة؛

publicBankAccount(string theOwner)

publicBankAccount(سلسلة theOwner)

Example of a Constructor (parameterized Constructor)

- If you put parameters in the constructor, then, when initializing the object, the parameters need to be passed, as shown in the following code:

```
BankAccount account = new BankAccount("Some Person");
```

Example of a Constructor (parameterized Constructor)

مثال على المنشئ (منشئ ذو معلمات)

 If you put parameters in the constructor, then, when initializing the object, the parameters need to be passed, as shown in the following code:

 إذا قمت بوضع معلمات في المُنشئ، فعند تهيئة الكائن، يجب تمرير المعلمات، كما هو موضح في الكود التالي:

```
BankAccount account = new BankAccount("Some Person");
```

```
حساب BankAccount = حساب BankAccount جديد("شخص ما");
```

Multiple Constructors in a class

- Another interesting thing is that you can have multiple constructors in a class.
- You might have one constructor that takes one argument and another that doesn't take any arguments.
- Depending on the way in which you are initializing the object, the respective constructor will be called. Let's look at the following example:

Multiple Constructors in a class

منشئون متعددون في الفصل

Another interesting thing is that you can have multiple constructors in a class.

شيء آخر مثير للاهتمام هو أنه يمكن أن يكون لديك عدة مُنشئين في الفصل الدراسي.

You might have one constructor that takes one argument and another that doesn't take any arguments.

قد يكون لديك مُنشئ واحد يأخذ وسيطة واحدة وآخر لا يأخذ أي وسيطات.

Depending on the way in which you are initializing the object, the respective constructor will be

اعتمادًا على الطريقة التي تقوم بها بتهيئة الكائن، سيكون المُنشئ المعني

called. Let's look at the following example:

مُسَمَّى. لننظر إلى المثال التالي:

Multiple Constructors in a class

```
class BankAccount
{
    public string owner;
    public BankAccount()
    {
        owner = "Some person";
    }
    public BankAccount(string theOwner)
    {
        owner = theOwner;
    }
}
```

Multiple Constructors in a class

منشئون متعددون في الفصل

publicstring owner;

مالك السلسلة العامة؛

publicBankAccount()

حساب البنك العام ()

owner = "Some person";

```
المالك = "شخص ما";
```

publicBankAccount(string theOwner)

publicBankAccount(سلسلة theOwner)

Multiple Constructors in a class

- In the preceding example, we can see that we have two constructors for the BankAccount class.
- If you pass a parameter when you create a BankAccount object, it will call the second constructor, which will set the value and create the object.

Multiple Constructors in a class

منشئون متعددون في الفصل

In the preceding example, we can see that we have two constructors for the BankAccount class.

في المثال السابق، يمكننا أن نرى أن لدينا مُنشئين لفئة BankAccount.

If you pass a parameter when you create a BankAccount object, it will call the second constructor, which will set the value and create the object.

إذا قمت بتمرير معلمة عند إنشاء كائن BankAccount، فسيتم استدعاء المنشئ الثاني، الذي سيقوم بتعيين القيمة وإنشاء الكائن.

Multiple Constructors in a class

- If you don't pass a parameter while creating the object, the first constructor will be called. If you don't have the appropriate constructors, then the method of object creation will not work.

Multiple Constructors in a class

منشئون متعددون في الفصل

- If you don't pass a parameter while creating the object, the first constructor will be called. If you don't have the appropriate constructors, then the method of object creation will not work.
- إذا لم تقم بتمرير معلمة أثناء إنشاء الكائن، فسيتم استدعاء المنشئ الأول. إذا لم يكن لديك المُنشئين المناسبين، فلن تعمل طريقة إنشاء الكائن.

Implicit Constructors in a class

- If you don't create a constructor, then the compiler creates an empty constructor for that class, as follows:

```
class BankAccount
{
    public string owner;
    public BankAccount()
    {
    }
}
```

Implicit Constructors in a class

المنشئون الضمنيون في الفصل

 If you don't create a constructor, then the

 إذا لم تقم بإنشاء منشئ، ثم

compiler creates an empty constructor for that class, as follows:

يقوم المترجم بإنشاء مُنشئ فارغ لتلك الفئة، كما يلي:

publicstring owner;

صفحة (42)

publicBankAccount()

حساب البنك العام ()

Characteristics of OOP

- OOP is one of the most important programming methodologies nowadays. The whole concept depends on four main ideas, which are known as the **pillars of OOP**. These four pillars are as follows:
  - Inheritance
  - Encapsulation
  - Polymorphism
  - Abstraction

Characteristics of OOP

خصائص OOP

 OOP is one of the most important programming methodologies nowadays. The whole concept depends on four main ideas, which are known as the pillars of OOP. These four

 تعد OOP واحدة من أهم منهجيات البرمجة في الوقت الحاضر. يعتمد المفهوم بأكمله على أربعة أفكار رئيسية، والتي تُعرف باسم ركائز OOP. هذه الركائز الأربع هي كما

follows:

يتبع:

- Inheritance
- ميراث
- Encapsulation
- التغليف
- Polymorphism
- تعدد الأشكال
- Abstraction

- التجريد

صفحة (43)

Inheritance

- The word inheritance means receiving or deriving something from something else. In real life, we might talk about a child inheriting a house from his or her parents. In that case, the child has the same power over the house that his parents had

Inheritance

ميراث

 The word inheritance means receiving or

 كلمة الميراث تعني التلقي أو

deriving something from something else. In real life, we might talk about a child inheriting a house from his or her parents. In that case, the child has the same power

يستنتج شيئا من شيء آخر. في الحياة الواقعية، قد نتحدث عن طفل يرث منزلاً من والديه. وفي هذه الحالة، يكون للطفل نفس السلطة على المنزل التي كانت لوالديه

Inheritance

- This concept of inheritance is one of the pillars of OOP.
- In programming, when one class is derived from another class, this is called inheritance. This means that the derived class will have the same properties as the parent class.

Inheritance

ميراث

This concept of inheritance is one of the pillars of OOP.

يعد مفهوم الميراث هذا أحد ركائز OOP.

In programming, when one class is derived from another class, this is called inheritance. This means that the derived class will have the same properties as the

في البرمجة، عندما يتم اشتقاق فئة واحدة من فئة أخرى، وهذا ما يسمى الميراث. هذا يعني أن الفئة المشتقة سيكون لها نفس خصائص الفئة الأصلية.

Inheritance

- In programming terminology, the class from which another class is derived is called the **parent class**, while the classes that inherit from these are called **child classes**.

Inheritance

ميراث

- In programming terminology, the class from which another class is derived is called the parent class, while the classes that inherit from these are called child classes.
- في مصطلحات البرمجة، تسمى الفئة التي يشتق منها فئة أخرى الفئة الأصل، بينما تسمى الفئات التي ترث منها الفئات الفرعية.

Inheritance Example

```
public class Fruit
{
    public string Name { get; set; }
    public string Color { get; set; }
}
public class Apple : Fruit
{
    public int NumberOfSeeds { get; set; }
}
```

Inheritance Example

مثال الميراث

publicstring Name { get; set; }

اسم السلسلة العامة {احصل على؛ تعيين؛ }

publicstring Color { get; set; }

لون السلسلة العامة {احصل على؛ تعيين؛ }

publicclassApple : Fruit

publicclassApple: الفاكهة

صفحة (47)

publicint NumberOfSeeds { get; set; } }

publicint NumberOfSeeds { get; تعيين؛ } {

Inheritance Example

- In the preceding example, we used inheritance. We have a parent class, called Fruit. This class holds the common properties that every fruit has: a **Name** and a **Color**. We can use this Fruit class for all fruits.

- If we create a new class, called Apple, this class can inherit the Fruit class because we know that an apple is a fruit.

Inheritance Example

مثال الميراث

In the preceding example, we used inheritance. We have a parent class, called Fruit. This class holds the common

في المثال السابق، استخدمنا الميراث. لدينا فئة الوالدين، تسمى الفاكهة. هذه الفئة تحمل المشتركة

properties that every fruit has: a Name and a Color. We can use this Fruit class for all fruits.

الخصائص التي تمتلكها كل فاكهة: الاسم واللون. يمكننا استخدام فئة الفاكهة هذه لجميع الفواكه.

If we create a new class, called Apple, this class can inherit the Fruit class because we know that an apple is a fruit.

إذا قمنا بإنشاء فئة جديدة، تسمى Apple، فيمكن لهذه الفئة أن ترث فئة الفاكهة لأننا نعلم أن التفاحة هي فاكهة.

Inheritance Example

- The properties of the Fruit class are also properties of the Apple class.

- If the Apple inherits the Fruit class, we don't need to write the same properties for the Apple class because it inherits these from the Fruit class.

Inheritance Example

مثال الميراث

The properties of the Fruit class are also properties of the Apple class.

خصائص فئة الفاكهة هي أيضًا خصائص فئة Apple.

If the Apple inherits the Fruit class, we don't need to write the same properties for the Apple class because it inherits these from the Fruit class.

إذا ورثت Apple فئة الفاكهة، فلن نحتاج إلى كتابة نفس الخصائص لفئة Apple لأنها ترث هذه الخصائص من فئة الفاكهة.

Encapsulation

- Encapsulation means hiding or covering. In C#, encapsulation is achieved by access modifiers. The access modifiers that are available in C# are the following:
  - Public
  - Private
  - Protected
  - Internal
  - Internal protected

Encapsulation

التغليف

Encapsulation means hiding or covering. In C#, encapsulation is achieved by access modifiers. The access modifiers that are available in C# are the following:

التغليف يعني الإخفاء أو التغطية. في C#، يتم التغليف عن طريق معدلات الوصول. معدّلات الوصول المتوفرة في C# هي كما يلي:

- Public

- عام

- Private
- خاص
- Protected
- محمي
- Internal
- داخلي
- Internal protected

- محمية داخلية

صفحة (50)

Encapsulation

- Encapsulation is when you want to control other classes' access to a certain class. Let's say you have a BankAccount class. For security reasons, it isn't a good idea to make that class accessible to all classes. It's better to make it Private or use another kind of access specifier.

Encapsulation

التغليف

Encapsulation is when you want to control other classes' access to a certain class. Let's say you have a BankAccount class. For security reasons, it isn't a good idea to make

يتم التغليف عندما تريد التحكم في وصول الفئات الأخرى إلى فئة معينة. لنفترض أن لديك فئة BankAccount. لأسباب أمنية، ليس من الجيد إنشاء هذا الفصل

accessible to all classes. It's better to make it Private or use another kind of access specifier.

في متناول جميع الطبقات. من الأفضل جعله خاصًا أو استخدام نوع آخر من محددات الوصول.

Encapsulation

- You can also limit access to the properties and variables of a class. For example, you might need to keep the BankAccount class public for some reason, but make the AccountBalance property private so that no other class can access this property except the BankAccount class

- Like variables and properties, you can also use access specifiers for methods. You can write private methods that are not needed by other classes, or that you don't want to expose to other classes

Encapsulation

التغليف

You can also limit access to the properties and variables of a class. For example, you might need to keep the BankAccount class public for some reason, but make the AccountBalance property private so that no other class can access this property except

يمكنك أيضًا تقييد الوصول إلى خصائص ومتغيرات الفصل الدراسي. على سبيل المثال، قد تحتاج إلى الاحتفاظ بفئة BankAccount عامة لسبب ما، ولكن مع جعل خاصية AccountBalance خاصة بحيث لا تتمكن أي فئة أخرى من الوصول إلى هذه الخاصية باستثناء فئة BankAccount

Like variables and properties, you can also use access

مثل المتغيرات والخصائص، يمكنك أيضًا استخدام الوصول

specifiers for methods. You can write private methods that are not needed by other classes, or that you don't want to expose to other classes

محددات للطرق. يمكنك كتابة أساليب خاصة لا تحتاجها الفئات الأخرى، أو التي لا تريد كشفها للفئات الأخرى

Abstraction

- If something is abstract, it means that it doesn't have an instance in reality but does exist as an idea or concept.
- In programming, we use this technique to organize our thoughts. This is one of the pillars of OOP.
- In C#, we have **abstract classes**, which implement the concept of abstraction. Abstract classes are classes that don't have any instances
- classes that implement the abstract class will implement the properties and methods of that abstract class.

Abstraction

التجريد

If something is abstract, it means that it doesn't have an instance in reality but does exist as an idea or concept.

إذا كان الشيء مجردًا، فهذا يعني أنه ليس له مثيل في الواقع ولكنه موجود كفكرة أو مفهوم.

In programming, we use this technique to organize our thoughts. This is one of the pillars of OOP.

في البرمجة، نستخدم هذه التقنية لتنظيم أفكارنا. وهذا أحد ركائز OOP.

In C#, we have abstract classes, which implement the concept of abstraction. Abstract classes are classes that don't have any instances

في C#، لدينا فئات مجردة، والتي تنفذ مفهوم التجريد. الفئات المجردة هي فئات لا تحتوي على أي مثيلات

classes that implement the abstract class will implement the properties and methods of that abstract class.

ستقوم الفئات التي تنفذ الفئة المجردة بتنفيذ خصائص وأساليب تلك الفئة المجردة.

Abstraction Example

```
public abstract class Vehicle
{
    public abstract int GetNumberOfTyres();
}
public class Bicycle : Vehicle
{
    public string Company { get; set; }
    public string Model { get; set; }
    public int NumberOfTyres { get; set; }
    public override int GetNumberOfTyres()
    {
        return NumberOfTyres;
    }
}
```

Abstraction Example

مثال التجريد

publicclassBicycle : Vehicle

publicclassدراجة: مركبة

publicstring Company { get; set; }

publicstring شركة {احصل على؛ تعيين؛ }

publicstring Model { get; set; }

نموذج السلسلة العامة {احصل على؛ تعيين؛ }

صفحة (54)

```
publicint NumberOfTyres { get; set; }
```

```
publicint NumberOfTyres { get; تعيين؛ }
```

```
return NumberOfTyres;
```

```
إرجاع NumberOfTyres؛
```

Abstraction Example

- In the preceding example, we have an abstract class called Vehicle.
- It has one abstract method, called GetNumberOfTyres().
- As it is an abstract method, this has to be **overridden** by the classes that implement the abstract class.

Abstraction Example

مثال التجريد

In the preceding example, we have an abstract class called Vehicle.

في المثال السابق، لدينا فئة مجردة تسمى Vehicle.

It has one abstract method, called GetNumberOfTyres().

يحتوي على أسلوب مجرد واحد يسمى GetNumberOfTyres().

As it is an abstract method, this has to be overridden by the classes that implement the

نظرًا لأنها طريقة مجردة، يجب تجاوزها بواسطة الفئات التي تنفذ الطريقة

abstract class.

فئة مجردة.

Abstraction Example

- Our Bicycle and Car classes implement the Vehicle abstract class, so they also override the abstract method GetNumberOfTyres().

- If you take a look at the implementation of these methods in the two classes, you will see that the implementation is different, which is due to abstraction.

Abstraction Example

مثال التجريد

Our Bicycle and Car classes implement the Vehicle abstract class, so they also override the abstract method GetNumberOfTyres().

تطبق فئتا الدراجات والسيارات لدينا فئة مجردة للمركبة، لذا فهي تتجاوز أيضًا الطريقة المجردة GetNumberOfTyres().

If you take a look at the implementation of these methods in the two classes, you will see that the implementation is different, which is due to

إذا ألقيت نظرة على تنفيذ هذه الطرق في الفئتين، سترى أن التنفيذ مختلف، وذلك بسبب

abstraction.

التجريد.

Polymorphism

- The word polymorph means many forms.

- To understand the concept of polymorphism properly, let's work with an example. Let's think about a person, such as Bill Gates. We all know that Bill Gates is a great software developer, businessman, and philanthropist. He is one individual, but he has different roles and performs different tasks. This is polymorphism.

Polymorphism

تعدد الأشكال

The word polymorph means many forms.

كلمة متعدد الأشكال تعني عدة أشكال.

To understand the concept of polymorphism properly, let's work with an example. Let's think about a person, such as Bill Gates. We all know that Bill Gates is a great software developer, businessman, and philanthropist. He is one individual, but he has different

لفهم مفهوم تعدد الأشكال بشكل صحيح، دعونا نعمل مع مثال. دعونا نفكر في شخص مثل بيل جيتس. نعلم جميعًا أن بيل جيتس هو مطور برامج عظيم ورجل أعمال ومحسن. فهو فرد واحد، ولكن له أدوار وأدوار مختلفة

performs different tasks. This is polymorphism.

يؤدي مهام مختلفة. هذا هو تعدد الأشكال.

Polymorphism

- When Bill Gates was developing software, he was playing the role of a software developer. He was thinking about the code he was writing.
- Later, when he became the CEO of Microsoft, he started managing people and thinking about growing the business.
- He's the same person, but with different roles and different responsibilities.

Polymorphism

تعدد الأشكال

When Bill Gates was developing software, he was playing the role of a software developer. He was thinking about the code he was writing.

عندما كان بيل جيتس يطور البرمجيات، كان يلعب دور مطور البرمجيات. كان يفكر في الكود الذي كان يكتبه.

Later, when he became the CEO of Microsoft, he started managing people and thinking about growing the business.

وفي وقت لاحق، عندما أصبح الرئيس التنفيذي لشركة مايكروسوفت، بدأ في إدارة الأفراد والتفكير في تنمية الأعمال التجارية.

He's the same person, but with different roles and different responsibilities.

إنه نفس الشخص، ولكن بأدوار مختلفة ومسؤوليات مختلفة.

Polymorphism

- In C#, there are two kind of polymorphism: **static polymorphism** and **dynamic polymorphism**.

- **Static polymorphism** is a kind of polymorphism where the role of a method is determined at compilation time.
- in **dynamic polymorphism**, the role of a method is determined at runtime.

Polymorphism

تعدد الأشكال

In C#, there are two kind of polymorphism: static polymorphism and dynamic polymorphism.

يوجد في لغة C# نوعان من تعدد الأشكال: تعدد الأشكال الساكن وتعدد الأشكال الديناميكي.

Static polymorphism is a kind of polymorphism where the role of a method is determined at compilation time.

تعدد الأشكال الثابت هو نوع من تعدد الأشكال حيث يتم تحديد دور الطريقة في وقت التجميع.

in dynamic polymorphism, the role of a method is determined at runtime.

في تعدد الأشكال الديناميكي، يتم تحديد دور الطريقة في وقت التشغيل.

Polymorphism

- Examples of static polymorphism include method overloading and operator overloading.
- Let's take a look at an example of method overloading:

Polymorphism

تعدد الأشكال

Examples of static polymorphism include

تتضمن أمثلة تعدد الأشكال الساكنة

method overloading and operator overloading.

طريقة التحميل الزائد والتحميل الزائد للمشغل.

Let's take a look at an example of method overloading:

دعونا نلقي نظرة على مثال على التحميل الزائد للطريقة:

صفحة (60)

static polymorphism (method overloading)

```
public class Calculator
{
    public int AddNumbers(int firstNum, int secondNum)
    {
        return firstNum + secondNum;
    }
    public double AddNumbers(double firstNum, double secondNum)
    {
        return firstNum + secondNum;
    }
}
```

static polymorphism (method overloading)

تعدد الأشكال الساكن (طريقة التحميل الزائد)

return firstNum + secondNum;

إرجاع الرقم الأول + الرقم الثاني؛

publicdouble AddNumbers(double firstNum, double secondNum) {

publicdouble AddNumbers(الرقم الأول المزدوج، الرقم الثاني المزدوج) {

static polymorphism (method overloading)

- we can see that we have two methods with the same name, **AddNumbers**.

- Normally, we can't have two methods that have the same name; however, as the parameters of those methods are different, methods are allowed to have the same name by t he compiler.

static polymorphism (method overloading)

تعدد الأشكال الساكن (طريقة التحميل الزائد)

we can see that we have two methods with the same name, AddNumbers.

يمكننا أن نرى أن لدينا طريقتين بنفس الاسم، AddNumbers.

Normally, we can't have two methods that have the same name; however, as the parameters of those methods are different, methods are

في العادة، لا يمكن أن يكون لدينا طريقتان لهما نفس الاسم؛ ومع ذلك، بما أن معلمات تلك الأساليب مختلفة، فإن الأساليب مختلفة

allowed to have the same name by t he compiler.

مسموح له أن يكون له نفس الاسم بواسطة المترجم.

static polymorphism (method overloading)

- Writing a method with the same name as another method, but with different parameters, is called **method overloading**. This is a kind of **polymorphism**.

static polymorphism (method overloading)

تعدد الأشكال الساكن (طريقة التحميل الزائد)

 Writing a method with the same name as

 كتابة طريقة بنفس الاسم

another method, but with different parameters, is called method overloading. This is a kind of

طريقة أخرى، ولكن مع معلمات مختلفة، تسمى طريقة التحميل الزائد. هذا نوع من

polymorphism.

تعدد الأشكال.

Dynamic polymorphism

- Dynamic polymorphism refers to the use of the abstract class.
- When you write an abstract class, no instance can be created from that abstract class.
- When any other class uses or implements that abstract class, the class also has to implement the abstract methods of that abstract class.

Dynamic polymorphism

تعدد الأشكال الديناميكي

Dynamic polymorphism refers to the use of the abstract class.

يشير تعدد الأشكال الديناميكي إلى استخدام الفئة المجردة.

When you write an abstract class, no instance can be created from that abstract class.

عندما تكتب فئة مجردة، لا يمكن إنشاء أي مثيل من تلك الفئة المجردة.

When any other class uses or implements that abstract class, the class also has to implement the abstract methods of that abstract class.

عندما تستخدم أي فئة أخرى تلك الفئة المجردة أو تنفذها، يجب على الفئة أيضًا تنفيذ الأساليب المجردة لتلك الفئة المجردة.

Dynamic polymorphism

- As different classes can implement the abstract class and can have different implementations of abstract methods, polymorphic behavior is achieved.

- In this case, we have methods with the same name but different implementations.

Dynamic polymorphism

تعدد الأشكال الديناميكي

As different classes can implement the abstract class and can have different implementations of abstract methods, polymorphic behavior is

نظرًا لأن الفئات المختلفة يمكنها تنفيذ الفصل المجرد ويمكن أن يكون لها تطبيقات مختلفة للطرق المجردة، فإن السلوك متعدد الأشكال هو كذلك

achieved.

حقق.

In this case, we have methods with the same name but different implementations.

في هذه الحالة، لدينا طرق بنفس الاسم ولكن التنفيذ مختلف.

static polymorphism (operator overloading)

- Like method overloading, **operator overloading** is also a **static polymorphism**. Let's look at an example of operator overloading to demonstrate this:

static polymorphism (operator overloading)

تعدد الأشكال الساكن (التحميل الزائد على المشغل)

 Like method overloading, operator overloading is also a static polymorphism. Let's look at an example of operator overloading to demonstrate this:

 مثل التحميل الزائد للطريقة، فإن التحميل الزائد للمشغل هو أيضًا تعدد أشكال ثابت. دعونا نلقي نظرة على مثال التحميل الزائد للمشغل لتوضيح ذلك:

static polymorphism (operator overloading)

```
public class MyCalc
{
    public int a;
    public int b;
    public MyCalc(int a, int b)
    {
        this.a = a;
        this.b = b;
    }
    public static MyCalc operator +(MyCalc a, MyCalc b)
    {
        return new MyCalc(a.a * 3, b.b * 3);
    }
}
```

static polymorphism (operator overloading)

تعدد الأشكال الساكن (التحميل الزائد على المشغل)

publicint a;

نشر أ؛

publicint b;

publicint ب؛

publicstatic MyCalc operator +(MyCalc a, MyCalc b)

عامل تشغيل MyCalc publicstatic +(MyCalc a، MyCalc b)

static polymorphism (operator overloading)

- In the preceding example, we can see that the plus sign (+) is overloaded with another kind of calculation. So if you sum up two MyCalc objects, you will get an overloaded result instead of the normal sum, and this overloading happens at compile time, so it is **static polymorphism**.

static polymorphism (operator overloading)

تعدد الأشكال الساكن (التحميل الزائد على المشغل)

 In the preceding example, we can see that the plus sign (+) is overloaded with another kind of calculation. So if you sum up two MyCalc

 في المثال السابق، يمكننا أن نرى أن علامة الجمع (+) مثقلة بنوع آخر من العمليات الحسابية. لذلك إذا قمت بتلخيص اثنين من MyCalc

objects, you will get an overloaded result instead of the normal sum, and this overloading

الكائنات، سوف تحصل على نتيجة مثقلة بدلا من المجموع العادي، وهذا التحميل الزائد

happens at compile time, so it is static

يحدث في وقت الترجمة، لذلك فهو ثابت

polymorphism.

تعدد الأشكال.