Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Four Pillars Of OOP

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Four Fundamentals of OOP

- The Four Fundamentals of OOP are:
  1. Abstraction
  2. Inheritance
  3. Encapsulation
  4. Polymorphism

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Encapsulation

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

ENCAPSULATION

- "Encapsulate" means to enclose something within a capsule or container.
- The "-tion" suffix is used to indicate the action or process of something.
- So, "encapsulation" refers to the process of enclosing or containing something within an object in programming.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Encapsulation

- Encapsulation is the containment and the grouping of our data members and member functions in a class and an object.
- We use to make our software more manageable, encapsulation reduce bugs, and organize our code
- Think about medicine or fish oil. They are put into capsules to prevent the compounds from being mixed together. This is what encapsulation does.
- Encapsulation is a product of using classes.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Encapsulation Vs Data Hiding

- Encapsulation it is commonly thought to be the process of “Data or Information Hiding”, but that is a common misconception. Data hiding is a feature of encapsulation but you can encapsulate without hiding data
- Think about the fish oil capsule. You can see the fish oil, so it is not hidden but it is still encapsulated

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Data Hiding

- Data hiding is where we set the level of accessibility an objects properties and functions have.
- You as a human have an age, I can't change your age or even know your age without the proper security clearance. This is an encapsulated property you have that is hidden from the world.
- A game enemy has a numerical location. As the player you cannot see the enemy numerical location. This data is hidden.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Access Modifiers

- Now we know Encapsulation is a strategy in object-oriented programming that allows us to keep an object's inner details hidden and neatly arranged. To do this, we use a process called "Data hiding" or "Information hiding," which is an essential part of encapsulation. This process involves concealing an object's information and granting access only through designated methods. Access modifiers aid in data hiding by determining who can interact with an object's properties and actions. In this lesson, we'll examine the different access modifiers in OOP and discover how they strengthen data hiding and encapsulation.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Access Modifiers

- Access: This refers to the ability to use or interact with something. In the context of programming, access usually means being able to read or modify data or execute certain functions or methods.
- Modifiers: These are tools or mechanisms that change the behavior of something. In programming, modifiers can be used to change the access level or visibility of data and functions, among other things.
- Access modifiers: Therefore, access modifiers in object-oriented programming are tools that control how and where different parts of a program can access or use an object's properties and methods. They are used to limit or grant access to data and functions, depending on the needs of the program and the design of the object.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Access Modifiers

- Imagine a house with different rooms, each containing valuable items. This house represents an object, and the rooms represent the properties and methods of the object. Access modifiers determine who can enter each room.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Public Access Modifier

- The public access modifier is used to declare a property or method that can be accessed from anywhere in the program. In our imaginary house, the doors to all the rooms are wide open and anyone can enter any room. This means that any part of the program can access the object's properties and methods, just like anyone can walk into a public museum and look at the exhibits without any restrictions.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Private Access Modifier

- The private access modifier is used to declare a property or method that can only be accessed within the same class. In our imaginary house, the doors to some of the rooms are locked, and only members of the family (and close friends) have keys to these rooms. This means that only the object itself can access its private properties and methods, just like only the owner of a personal safe can access its contents.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Protected Access Modifier

- The protected access modifier is used to declare a property or method that can be accessed within the same class and its subclasses. In our imaginary house, the doors to some of the rooms are locked, but members of the same household have keys to these rooms. This means that the object and its subclasses can access its protected properties and methods, just like only family members can access certain areas of a shared living space.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Default Access Modifier

- The default access modifier, also known as package-private, is used to declare a property or method that can be accessed within the same package. In our imaginary house, the doors to some of the rooms are locked, and only the residents of the same package have keys to these rooms. This means that only classes within the same package can access the object's default properties and methods, just like only residents of a gated community can access its amenities.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Access Modifiers - Conclusion

- Access modifiers are an important part of OOP that allow us to control how an object's properties and methods can be accessed from different parts of a program. By understanding the different types of access modifiers, we can design our programs to be more secure, efficient, and maintainable. As beginners in OOP, it's essential to grasp these concepts early on, as they form the foundation of object-oriented programming.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Setters and Getters

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Getters and Setters

- When we encapsulate and make data members or fields private, we keep them hidden from outside the object.
- Getters and setters help us modify an object's data. For instance, the ".text" getter and setter in "Textbox1.text" lets us change or retrieve the textbox's text value.
- Getters retrieve data from an object
- Setters set data in an object
- Getters are also called Accessors
- Setters are also called Mutators

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Getters and Setters

- Getters and Setters are a feature of encapsulation, and can also contribute to abstraction (which we will talk about later).
- When we encapsulate and make data members private, the objects data members can’t be accessed from the outside of the object.
- Making data members public can expose them to unintended modifications and potential errors.
- Getters and setters allow us to get information from the user and process the data without expectations from the user
- Let's say you want to get some data from a user about their age, and they input their age as -1 or 500. This could cause issues. By using getters and setters, we can process the data based on the parameters we define.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Getters and Setters

- Getters and setters, when used correctly, support encapsulation. However, if not used carefully, they can be like taking a tiny needle and puncturing our fish oil capsule, causing the fish oil to leak out.
  - To prevent leaks, be mindful when using getters and setters:
  - Don't create getters and setters for every data member, as this could lead to vulnerabilities and expose the data you want to protect.
  - Be aware that getters can reveal hidden data and setters can introduce unwanted modifications.
  - Implement logic within your methods to handle data processing for the user, rather than expecting the user to provide the correct input or perform the necessary processing.
- In summary, getters and setters are useful when applied thoughtfully, ensuring that the fish oil capsule of encapsulation remains intact and providing a clean interface for users.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Take aways from Encapsulation

- Encapsulation combines data and functions, like how a fish oil capsule keeps fish oil contained within its container. It helps keep code organized and prevents accidental changes.
- Data hiding protects an object's sensitive information and only exposes necessary details. This is similar to how an opaque or “non-see-through” capsule keeps the contents hidden from view.
- Access modifiers control which parts of the program can use and see an object's properties and methods. This helps enforce data hiding and prevent unauthorized access.
- By understanding encapsulation, data hiding, and access modifiers, developers can write better code that is more organized, secure, and maintainable.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance and Composition

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance and Composition

- We are now going to talk about:

1) Inheritance
2) Object Composition

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance and Composition

- As an overview, inheritance and object composition are two ways of inheriting data in object-oriented programming. Inheritance is like creating a new cake recipe based on an existing one, while object composition is like building a cake by combining different pre-made cake layers and toppings. Later, we will take a closer look at each concept.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance

- Inherit: to receive or get qualities, characteristics, or properties from something that came before.
- Inheritance: a way for a new class (a "subclass" or "derived class") to get the properties and methods of an existing class (a "parent" or "base class").

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance

- Inheritance is a powerful technique in object-oriented programming. It allows us to create a hierarchy of classes and have child classes inherit properties and methods from their parent class. By doing this, we can avoid rewriting code that already exists, which saves time and effort.

- To help visualize this concept, we can think of it as a family tree, where the parent class is at the top and the child classes are below it. Just like how children inherit traits from their parents, child classes inherit properties and methods from their parent class. This makes our code easier to organize and maintain.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance

- We can also think of inheritance like building with legos. The base class is like a lego block with common properties and methods. The child classes are like lego blocks that snap onto the base block and add their own unique properties and methods on top of it. This makes our code more flexible and modular

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance

- An example of inheritance is when we have a dog class that inherits some of its properties and functions from the animal class. This is because all animals share some common characteristics, such as eating, sleeping, and aging.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance

- The way a class inherits from another is by using a colon:

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

INHERITANCE - SAMPLE

```
public class Vehicle
{
    0 references
    public int NumWheels { get; set; }
    2 references
    public string Make { get; set; }
    2 references
    public string Model { get; set; }
    2 references
    public int Year { get; set; }
    1 reference
    public Vehicle(string make, string model, int year)
    {
        Make = make;
        Model = model;
        Year = year;
    }
    0 references
    public void Start()
    {
        MessageBox.Show("Vehicle started.");
    }
    1 reference
    public void Stop()
    {
       MessageBox.Show("Vehicle stopped.");
    }
}
```

```
public class Car : Vehicle
{
    1 reference
    public int NumDoors { get; set; }
    1 reference
    public Car(string make, string model, int year, int numDoors) : base(make, model, year)
    {
        this.NumDoors = numDoors;
    }
    1 reference
    public void Drive()
    {
        MessageBox.Show("Car is driving.");
    }
    0 references
    public void Brake()
    {
        MessageBox.Show("Car is braking.");
    }
}
```

```
Car myCar = new Car("Honda", "Civic", 2021, 4);
richTextBox1.Text ="Make: " + myCar.Make;
richTextBox1.Text ="Model: " + myCar.Model;
richTextBox1.Text ="Year: " + myCar.Year;
myCar.Drive();
myCar.Stop();
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance

- When creating an inherited class, accessing data members can be challenging.
- To address this, we can use the "protected" access modifier instead of "public".
- This allows data members to be accessed by child classes while still keeping them hidden from outside classes.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Inheritance

- Make an animal class and combine it with your class Dog and add a class Cat.
- An animal class gives its properties to a dog and cat class
- Some common characteristics, such as eating, sleeping, and aging.
- Move anything you think can be used in all animals from your Dog to the animal class

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Object Composition

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Object Composition

- Object composition is when you instantiate an object in another object
- Object composition is useful if you want to use characteristics from another object.
- Think about a customer and employee. Both of these have names, addresses, DOB
- By using an object that defines these general characteristics you can make

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Object Composition

- We will discuss this further in C++.
- But if you would like to experiment with it, you can split the Person class and the Address class and then compose the Address object within the Person object.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction - Defined

- "Abstract" means to consider something in a general or conceptual way, rather than in a specific or concrete way.
- The "-ion" suffix is used to indicate the act of doing something.
- So, "abstraction" refers to the act of considering something in a general or conceptual way, rather than focusing on specific details

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction - What

- In object-oriented programming, abstraction is a way to focus on what's important about an object, without worrying about how it works. This means we can work with objects in a general and conceptual way, instead of getting bogged down in specific details.
- Abstraction helps make our code more flexible and adaptable, because we can focus on the big picture of an object, rather than getting lost in the small details. This also helps us create code that can be used in multiple ways, because we can create abstract blueprints that can be used by different objects with their own unique features

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction – What

- Abstraction is a design concept and a process of implementation
- There are abstract classes that we can use that can not be instantiated and are used as high level classes which many other classes are inherited from

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction - Benefits

- Abstraction simplifies complexity by hiding irrelevant details for users.
- Users can interact with interfaces without needing to know their internal workings.
- Abstraction enables developers to create versatile, reusable code for various needs.
- Abstraction streamlines software development, making it easier to use and maintain.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction – Benefits

- With abstraction its better to have a simple interface with high abstraction vs a complex interface with minimal abstraction.
- Google search has a lot going on behind the scenes. But we only see a search bar and a few buttons

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction – Abstract Classes

- In C#, an abstract class is a class that cannot be instantiated on its own but can be used as a base class for other classes. An abstract class is designed to be inherited by other classes, and it typically contains one or more abstract methods, which are defined in the abstract class but do not have any implementation.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction – Abstract Classes

- In the context of the car class we created earlier, we can define an abstract class called "Vehicle" that contains the basic properties and methods of all vehicles. For example, we could define an abstract method called "Drive" in the Vehicle class that does not have an implementation, because the implementation will vary depending on the type of vehicle.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction – Abstract Classes

```
public abstract class Vehicle
{
    2 references
    public string Make { get; set; }
    2 references
    public string Model { get; set; }
    2 references
    public int Year { get; set; }
    1 reference
    public Vehicle(string make, string model, int year)
    {
        Make = make;
        Model = model;
        Year = year;
    }
    2 references
    public abstract void Drive();
    0 references
    public void Start()
    {
        MessageBox.Show("Vehicle started.");
    }
    1 reference
    public void Stop()
    {
       MessageBox.Show("Vehicle stopped.");
    }
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction – Abstract Classes

- In this example, the Vehicle class has three properties that are common to all vehicles: NumWheels, Model, and Year. It also has an abstract method called "Drive" that does not have an implementation.
- We can then create a Car class that inherits from the Vehicle class and provides an implementation for the Drive method:

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction – Abstract Classes

```
public class Car : Vehicle
{
    1 reference
    public int NumDoors { get; set; }
    1 reference
    public Car(string make, string model, int year, int numDoors) : base(make, model, year)
    {
        this.NumDoors = numDoors;
    }
    2 references
    public override void Drive()
    {
        MessageBox.Show("Car is driving.");
    }
    0 references
    public void Brake()
    {
        MessageBox.Show("Car is braking.");
    }
}
```

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction – Abstract Classes

- Abstract classes are useful when we want to define a common interface for a group of related classes, but we do not want to provide an implementation for all the methods in the base class. This allows us to enforce certain functionality in all the classes that inherit from the base class, while still allowing each class to provide its own unique implementation for certain methods.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Take aways from Abstraction

- Abstraction hides implementation from the end user
- Abstraction is different than Data hiding
- Turn the animal class into an abstract class
- Make the Eating function virtual override it calling the base class in both the Dog and Cat
- Make the Sleeping abstract override it with a new implementation in both the Dog and Cat.

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Abstraction Vs Encapsulation

- http://www.tonymarston.co.uk/php-mysql/abstraction.txt

Institute of
Technology
Development
of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism(more then one shape)

- "Poly": many or multiple
- "Morph": form or shape
- "Is": a verb indicating identity or equivalence
- So, "polymorphism" refers to the idea of having many or multiple forms or shapes that are all considered equivalent or have the same identity. In programming, this can refer to the ability of objects to take on multiple forms or to be used in multiple ways, depending on the context or situation.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism(more then one shape)

- Polymorphism in cars enables the creation of a single function, like "Drive," to be used by different types of cars, like sedans, SUVs, and sports cars.
- All cars can drive, but they may drive differently based on their type.
- With polymorphism, we can create a "Drive" function that can handle these differences and be used by all types of cars.
- Using polymorphism simplifies our code by not requiring separate "Drive" functions for each type of car.
- Polymorphism also makes our code more modular and maintainable.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism(more then one shape)

- Polymorphism is like a chameleon that can change its colors to blend in with its surroundings. Just like how a chameleon can take on different forms to suit its environment, polymorphism allows a method or variable to refer to different types of objects in programming.

INSTITUTE OF
TECHNOLOGY
DEVELOPMENT
OF CANADA

475 GRANVILLE STREET, VANCOUVER, BC, V6C 1T1
PHONE: +1(604)558-8727, +1(604)409-8200
TOLL FREE: +1(888) 880-4410
FAX: +1(888) 881-6545
WEB: WWW.ITDCANADA.CA
EMAIL: STUDYING@ITDCANADA.CA

POLYMORPHISM(MORE THEN ONE SHAPE)

- There are two types of polymorphism: compile-time polymorphism and runtime polymorphism. Compile-time polymorphism, also known as method overloading, is like having keys on a piano that produce different sounds when played. Although the keys may look the same, each key is programmed to produce a different sound when played. Similarly, in programming, different methods with the same name but different parameters can be used to perform different tasks.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism(more then one shape)

- Runtime polymorphism, also known as method overriding, is like a customized car that is built upon a basic car model. The basic car model provides a foundation for the customized car, but the customized car has unique features that make it different from the original model. Similarly, a subclass can provide a specific implementation of a method that is already provided by its parent class. The parent class provides the basic functionality, while the subclass adds unique features to make it different from the parent class. We are going to be focusing on this on for the duration of this class

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism(more then one shape)

- Polymorphism works by allowing a method to be defined in a base class and then overridden in a derived class. The method in the derived class has the same name and parameters as the method in the base class, but it provides a specific implementation for the derived class.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism – Binding

- Static binding and dynamic binding are two concepts related to polymorphism in programming.
- Static binding is like a bookshelf with fixed shelves that can only hold certain sizes of books. Just like how the shelves are designed to hold specific sizes of books, the compiler in static binding determines which method to use based on the reference type of the object.
- Dynamic binding, on the other hand, is like a shelf with adjustable heights that can hold any size of book. Just like how the adjustable shelf can accommodate different sizes of books, dynamic binding determines which method to use at runtime based on the actual type of the object.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism – Binding

- An analogy to further explain the difference is a restaurant menu. The static binding is like a set menu with fixed options that the chef can only prepare based on what is written on the menu. The customer must choose from the options listed, and the chef will prepare the food based on that choice. In dynamic binding, the customer can customize their order and ask the chef to prepare the food based on their preferences. The chef can adjust the recipe and cooking method based on the customer's request.

- In programming, static binding is faster and more efficient, but less flexible since the method to use is determined at compile time. Dynamic binding is more flexible but slower since the method to use is determined at runtime.

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism – Binding

- An example of dynamic binding is using a parent class as a generic type like in this code sample:

```
Vehicle[] vehicle = new Vehicle[3];
Car mazdaCX5 = new Car("Mazda", "CX5", 2022, 2, 5);
Car FordTaurus = new Car("Ford", "Taurus", 2024, 4, 13);
Truck DodgeRAM = new Truck("Dodge", "RAM", 2004, 2);
vehicle[0] = mazdaCX5;
vehicle[1] = FordTaurus;
vehicle[2] = DodgeRAM;
for (int i = 0; i < vehicle.Length; i++)
{
    richTextBox1.Text += vehicle[i].Drive() + "\n";
}
```

Institute of Technology Development of Canada

475 Granville Street, Vancouver, BC, V6C 1T1
Phone: +1(604)558-8727, +1(604)409-8200
Toll Free: +1(888) 880-4410
Fax: +1(888) 881-6545
Web: www.itdcanada.ca
Email: studying@itdcanada.ca

Polymorphism – Binding

- To Simplify this further we can do:

```
Vehicle[] vehicle = new Vehicle[3];
vehicle[0] = new Car("Mazda", "CX5", 2022, 2, 5);
vehicle[1] = new Car("Ford", "Taurus", 2024, 4, 13);
vehicle[2] = new Truck("Dodge", "RAM", 2004, 2);
for (int i = 0; i < vehicle.Length; i++)
{
    richTextBox1.Text += vehicle[i].Drive() + "\n";
}
```