Môn: Lập trình Hướng đối tượng (Object Oriented Programming)

Chương 4. Kế thừa và Đa hình trên Java

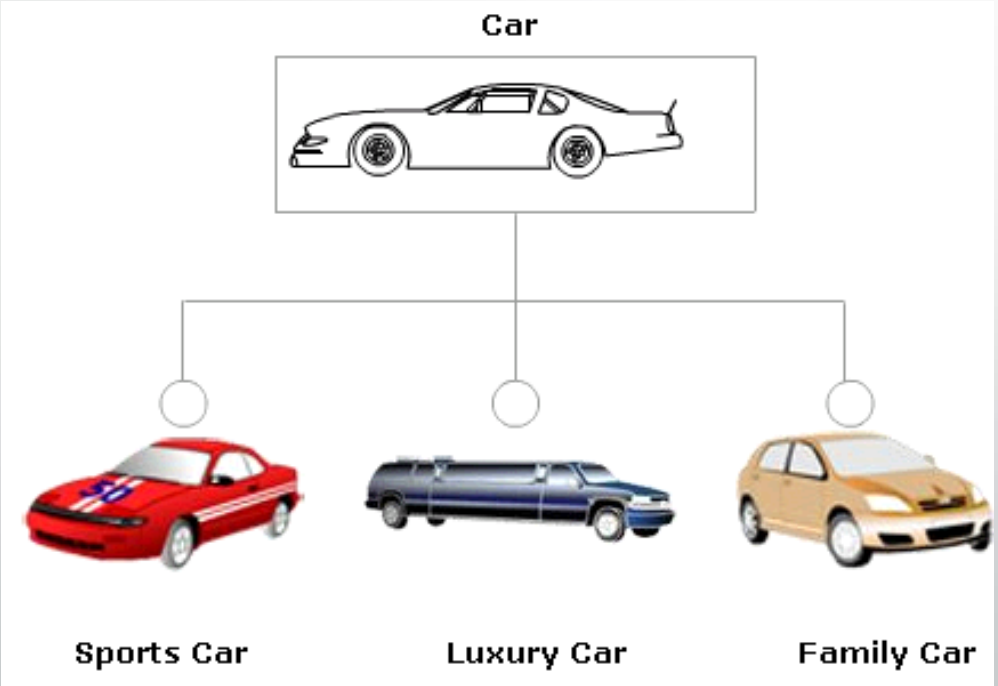
Nội dung

4.1. Kế thừa đơn (Single Inheritance)
4.2. Kế thừa kép (Multi-Inheritance)
4.3. Các lớp trừu tượng (Abtract Classes)
4.4. Interface
4.5. Đa hình (Polymorphism)
4.6. Case Study (Object Oriented Programs)
4.7. Một số lớp bản trong Java

4.1. Kế thừa đơn (Single Inheritance)

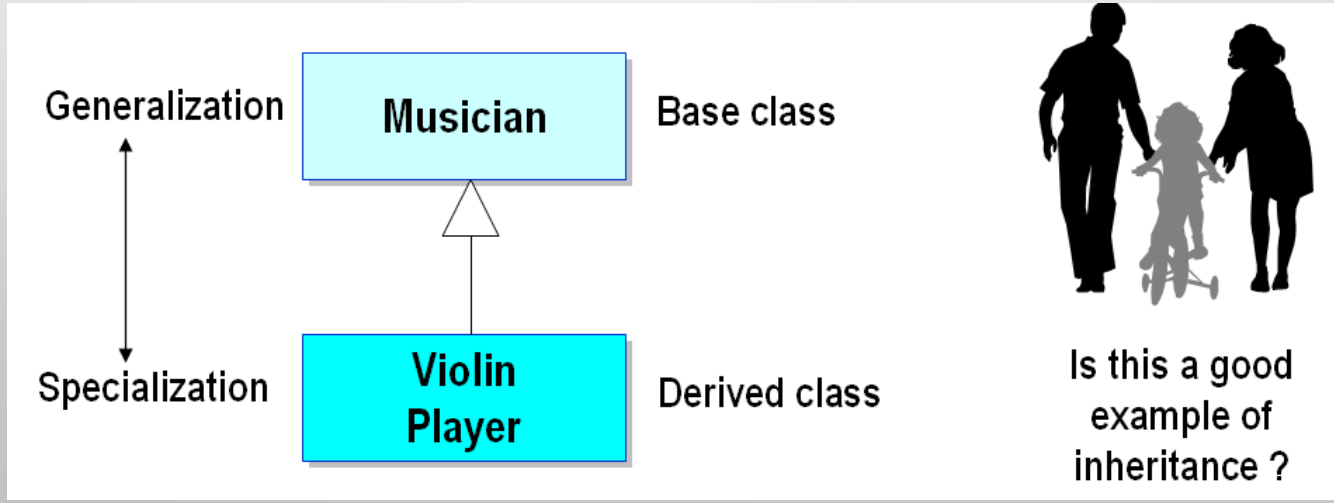
Thừa kế là gì?

- Tạo lớp mới từ một lớp đang tồn tại.
  - Sử dụng lại các trường (fields) và phương thức (methods)



4.1. Kế thừa đơn (tt)

- Lớp cha - Superclass
  - Lớp cho lớp khác thừa kế các trường và phương thức
  - Chúng được gọi là lớp cơ sở (base class) hoặc lớp cha (parent class)
- Lớp con - Subclass
  - Lớp được dẫn xuất (derive) từ lớp khác
  - Chúng được gọi là lớp dẫn xuất (derived class), lớp mở rộng (extended class) hoặc lớp con (child class)



4.1. Kế thừa đơn (tt)

Các khái niệm cơ bản trong thừa kế trong Java

- Sử dụng từ khóa "extends" để tạo lớp con.
- Một lớp chỉ có thể dẫn xuất trực tiếp từ 1 lớp khác – đơn thừa kế (single inheritance)
- Nếu lớp con không thừa kế từ lớp cha nào, mặc định xem nó thừa kế từ lớp cha tên là Object
- Phương thức khởi tạo (hàm dựng) không được thừa kế. Hàm dựng của lớp cha có thể được gọi từ lớp con
- Một lớp con có thể thừa kế tất cả các thành phần ("protected") của lớp cha.

4.1. Kế thừa đơn (tt)

- Cú pháp cho đơn thừa kế trong Java

```
public class derived-class-name extends base-class-name
{
// derived class methods extend and possibly override
// those of the base class
}
```

4.1. Kế thừa đơn (tt)

- Ví dụ thừa kế đơn trong Java

```
public class Car
{
    private int mileage;
    private String color;
    protected String make;

    public void accelerate()
    {
        System.out.println("Car is acceletating");
    }
}
```

```
public class LuxuryCar extends Car
{
    private String perks;
}
```

```
public class TestCar
{
    public static void main(String[] args)
    {
        Car c = new Car();
        LuxuryCar lCar = new LuxuryCar();

        c.accelerate();
        // this will call the inherited method accelerate
        lCar.accelerate();
    }
}
```

Output:

```
Car is Accelerating
Car is Accelerating
```

4.1. Kế thừa đơn (tt)

- Từ khóa “super”: Sử dụng để truy xuất các thành phần của lớp cha và hàm dựng của chúng từ lớp con
- Sự thừa kế trong hàm khởi tạo - Constructor Inheritance
  a. Khai báo về thừa kế trong hàm khởi tạo
  b. Chuỗi các hàm khởi tạo (Constructor Chaining)
  c. Các nguyên tắc của hàm khởi tạo (Rules)
  d. Gọi tường minh hàm khởi tạo của lớp cha

4.1. Kế thừa đơn (tt)

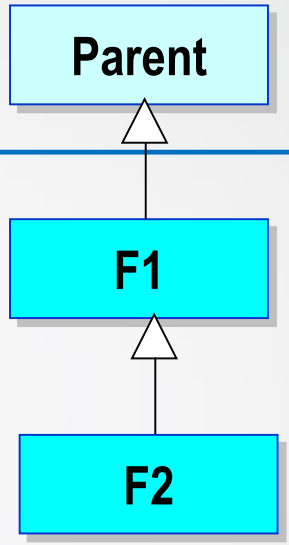
a. Khai báo về thừa kế trong hàm khởi tạo

- Trong Java, hàm khởi tạo không thể thừa kế từ lớp cha như các loại phương thức khác

- Khi tạo một thể hiện của lớp dẫn xuất , trước hết phải gọi đến hàm khởi tạo của lớp cha, tiếp đó mới là hàm khởi tạo của lớp con.

- Có thể gọi hàm khởi tạo của lớp cha bằng cách sử dụng từ khóa super trong phần khai báo hàm khởi tạo của lớp con.

4.1. Kế thừa đơn (tt)



b. Chuỗi các hàm khởi tạo (Constructor Chaining)

```
public class Parent
{
    public Parent()
    {
        System.out.println("Invoke parent default constructor");
    }
}
```

```
public class F1 extends Parent
{
    public F1()
    {
        System.out.println("Invoke F1 default constructor");
    }
}
```

```
public class F2 extends F1
{
    public F2()
    {
        System.out.println("Invoke F2 default constructor");
    }
}
```

```
public static void main(String [] args)
{
    new F2();
}
```

Khi tạo một thể hiện/đối tượng của lớp dẫn xuất (con), trước hết phải gọi đến hàm khởi tạo của lớp cha, tiếp đó là hàm khởi tạo của lớp con.

```
Invoke Parent default construct
Invoke F1 default constructor
Invoke F2 default constructor
```

1. Object
2. Parent() call super()
3. F1() call super ()
4. F2() call super()
5. Main() call new F2()

4.1. Kế thừa đơn (tt)

c. Các nguyên tắc của hàm khởi tạo (Rules)

- Hàm khởi tạo mặc định (default constructor) sẽ tự động sinh ra bởi trình biên dịch nếu lớp không khai báo hàm khởi tạo.
- Hàm khởi tạo mặc định luôn luôn không có tham số (no-arguments)

- Nếu trong lớp có định nghĩa hàm khởi tạo, hàm khởi tạo mặc định sẽ không còn được sử dụng.

- Nếu không có lời gọi tương minh đến hàm khởi tạo của lớp cha tại lớp con, trình biên dịch sẽ tự động chèn lời gọi tới hàm dựng mặc nhiên (implicity) hoặc hàm khởi tạo không tham số (explicity) của lớp cha trước khi thực thi đoạn code khác trong hàm khởi tạo lớp con.

4.1. Kế thừa đơn (tt)

d. Gọi tường minh hàm khởi tạo của lớp cha

```
public class Parent
{
    private int a;

    public Parent(int value)
    {
        a = value;
        System.out.println("Invoke parent parameter constructor");
    }
}
```

```
public class F1 extends Parent
{
    public F1(int value)
    {
        super (value);
        System.out.println("Invoke F1 default constructor");
    }
}
```

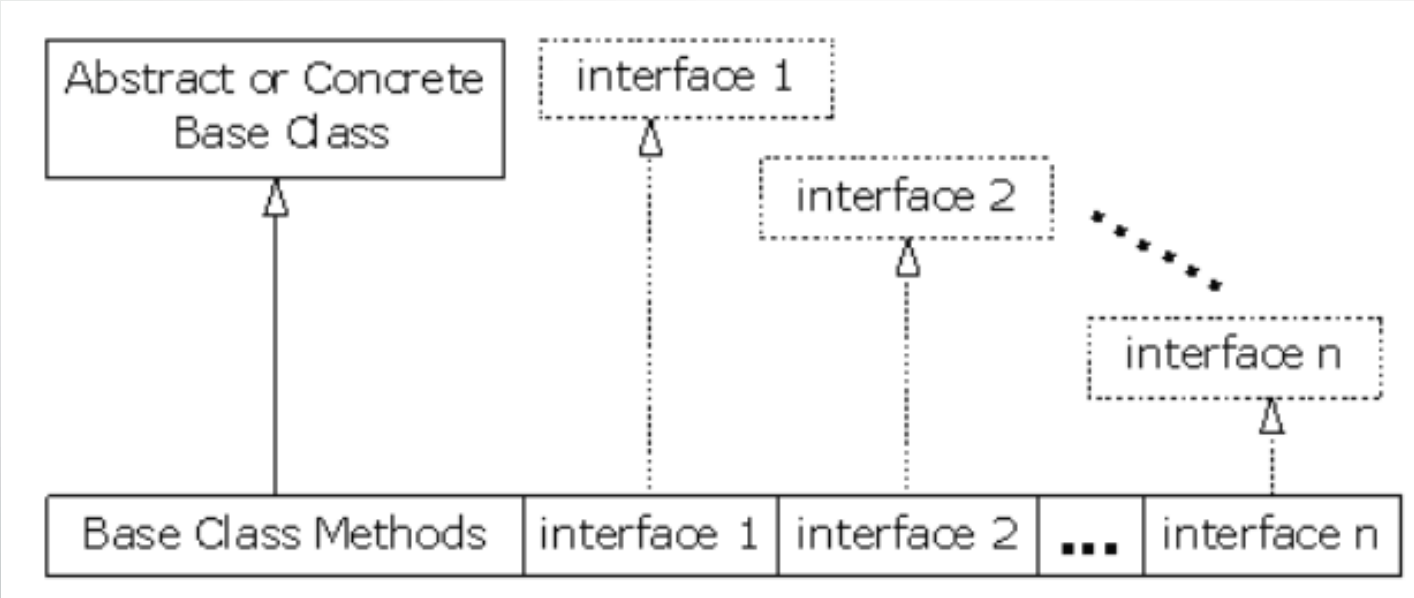
4.1. Kế thừa đơn (tt)

- Sử dụng truy cập protected trong thừa kế. The protected Access Modifier

Access Levels				
Modifier	**Class**	**Package**	**Subclass**	**World**
public	**Y**	**Y**	**Y**	**Y**
protected	**Y**	**Y**	**Y**	**N**
no modifier [package]	**Y**	**Y**	**N**	**N**
private	**Y**	**N**	**N**	**N**

4.2. Kế thừa kép (Multi-Inheritance)

- Java không cho phép đa kế thừa từ nhiều lớp cha/cơ sở
  - Đảm bảo tính dễ hiểu
  - Hạn chế xung đột
- Có thể cài đặt đồng thời nhiều giao diện



4.3. Lớp trừu tượng

- Có thể tạo ra các lớp cơ sở để tái sử dụng mà *không muốn tạo ra đối tượng thực của lớp*
  - Các lớp Point, Circle, Rectangle chung nhau khái niệm cùng là hình vẽ Shape → Giải pháp là khái báo lớp trừu tượng

- Lớp trừu tượng được xem như khung làm việc chung cung cấp các hành vi (behavior) cho các lớp khác.
- Không thể tạo đối tượng từ lớp trừu tượng
- Có thể thừa kế từ lớp trừu tượng
- Các lớp con phải hiện thực các phương thức trừu tượng được khai báo trong lớp trừu tượng (lớp cha).
- Khai báo lớp trừu tượng bằng cách sử dụng từ khóa abstract trước từ khóa class.

4.3. Lớp trừu tượng (tt)

- Cú pháp và ví dụ khai báo lớp trừu tượng

```
abstract class Shape
{
    protected int x, y;
    public Shape(int _x, int _y)
    {
        x = _x;
        y = _y;
    }
}
```

```
class Circle extends Shape
{
    int r;
    public Circle(int _x, int _y, int _r)
    {
        super(_x, _y);
        r = _r;
    }
}
```

```
class ShapTest
{
    public static void main(String[] args)
    {
        Shape s1 = new Circle(5, 5, 6);
        Shape s = new Shape(10, 10); // compile error
    }
}
```

4.3. Lớp trừu tượng (tt)

Phương thức trừu tượng

- Có thể khai báo các phương thức tại lớp cha/cơ sở nhưng được cài đặt thực tế tại lớp con/dẫn xuất
- Mỗi lớp con/dẫn xuất khác nhau có cách cài đặt khác nhau
- Phương thức trừu tượng bắt buộc phải định nghĩa lại tại lớp cha/dẫn xuất
- Là những phương thức chỉ có khai báo mà không có phần hiện thực.
  - Có từ khóa "abstract" trong phần khai báo phương thức
  - Phần khai báo sẽ không có cặp ngoặc và được kết thúc bởi dấu ; (semicolon)

4.3. Lớp trừu tượng (tt)

Lớp và Phương thức trừu tượng – Ví dụ

```
public abstract class Person
{
    protected int id;
    protected String name;

    public Person(int id, String name)
    {
        this.id = id;
        this.name = name;
    }

    public abstract void displayInfor();
}
```

```
public class Student extends Person
{
    public Student(int id, String name)
    {
        super(id, name);
    }

    public void displayInfor()
    {
        System.out.println("Student id: " + super.id);
        System.out.println("Student name: " + super.name);
    }
}
```

4.4. Interface

- Interface được định nghĩa như một kiểu tham chiếu và tương tự như lớp. Nó chứa một tập các quy tắc (các phương thức) mà các lớp cài đặt (hiện thực) phải tuân thủ.
- Interface chỉ có biến hằng, phương thức có dấu hiệu trừu tượng(abstract)
  - Các phương thức khai báo trong interface không bao gồm thân.
- Không thể khởi tạo đối tượng từ interface.
- Interface chỉ có thể được thừa kế từ các lớp hoặc các interface khác
- Một lớp khi hiện thực 1 interface (implements) cần phải hiện thực tất cả các phương thức của interface đó.

4.4. Interface (tt)

```
public interface UserDao {

    public final String TABLE ="User";

    public void create(User user);

    public void delete(int id);

    public void update(int id, User user);

}
```

```
public class UserDaoImpl implements UserDao {

    public void create(User user) {
        // ...
    }

    public void delete(int id) {
        // ...
    }

    public void update(int id, User user) {
        // ....
    }
}
```

4.4. Interface (tt)

Thực thi nhiều interfaces

- Một interface có thể thừa kế từ 1 hoặc nhiều interfaces khác
- Một lớp có thể thực thi nhiều interfaces.Quá trình thực thi này được coi như chức năng của đa thừa kế.
- Các interface được thực thi cách nhau bởi dấu phẩy khi khai báo thừa kế.
- Lớp con thừa kế phải thực thi tất cả các phương thức trừu tượng của interface.

```
public interface Quackable
{
    public void quack();
}
```

```
public interface Flyable
{
    public void fly();
}
```

```
public class RedheadDuck implements Quackable, Flyable
{
    public void quack()
    {
    }

    public void fly()
    {
    }
}
```

4.4. Interface (tt)

So sánh giữa lớp, lớp trừu tượng, interface

- Định nghĩa lớp bao gồm các biến lớp và các phương thức lớp , bao gồm cả phần signature và body
- Lớp trừu tượng bao gồm các biến, phương thức và tối thiểu 1 phương thức trừu tượng (phương thức không có phần chi tiết của hàm)
- Định nghĩa interface chỉ bao gồm phần signature của methods

4.5. Đa hình (Polymorphism)

- Java cung cấp 2 hình thức đa hình
  - Đa hình lúc biên dịch: Đa hình dạng tĩnh cho phép các phương thức cùng tên nhưng khác kiểu và tham số, Java xử lý bằng cách *overloading*
  - Đa hình lúc thực thi: Đa hình dạng động cho phép phương thức lớp con định nghĩa cụ thể các phương thức lớp cha, Java xử lý bằng cách *override*, phương thức lớp con có thể được gọi từ tham chiếu của lớp cha.

4.5. Đa hình (tt)

Static and dynamic binding

- Liên kết tĩnh: lời gọi hàm (phương thức) được quyết định khi biên dịch, do đó chỉ có một phiên bản của chương trình con được thực hiện → ưu điểm về tốc độ
- Liên kết động: lời gọi phương thức được quyết định khi thực hiện, phiên bản của phương thức phù hợp với đối tượng được gọi
  - Java mặc định sử dụng liên kết động

4.5. Đa hình (tt)

- Ví dụ đa hình - liên kết tĩnh

```
class OverloadingDemo
{
    public int add(int x, int y)
    { //method 1
        return x+y;
    }
    public int add(int x, int y, int z)
    { //method 2
        return x+y+z;
    }
    public int add(double x, int y)
    { //method 3
        return (int)x+y;
    }
}
```

4.5. Đa hình (tt)

- Ví dụ đa hình - liên kết động

```
class A
{
  void callme()
  {
      System.out.println("Inside A's callme method");
  }
}
```

```
class B extends A
{
  void callme()
  {
      System.out.println("Inside B's callme method");
  }
}
```

```
class C extends A
{
  void callme()
  {
      System.out.println("Inside C's callme method");
  }
}
```

```
class Dispatch
{
      public static void main(String args[])
      {
            A a = new A(); // object of type A
            B b = new B(); // object of type B
            C c = new C(); // object of type C
            A r; // obtain a reference of type A

            r = a; // r refers to an A object
            r.callme(); // calls A's version of callme

            r = b; // r refers to a B object
            r.callme(); // calls B's version of callme

            r = c; // r refers to a C object
            r.callme(); // calls C's version of callme
      }
}
```

4.5. Đa hình (tt)

Một số lưu ý khi override

- Danh sách tham số cần phải giống như danh sách tham số của phương thức bị override
- Kiểu trả về cần phải giống kiểu trả về của phương thức bị ghi đè ban đầu trong lớp cha.
- Mức truy cập không thể hạn chế hơn so với mức truy cập của phương thức bị ghi đè.
- Các phương thức được khai báo là final thì không được ghi đè.

4.6. Case Study

Công ty du lịch V quản lý thông tin là các chuyến xe. Thông tin của 2 loại chuyến xe:

- Chuyến xe nội thành: Mã số chuyến, Họ tên tài xế, số xe, số tuyến, số km đi được, doanh thu.
- Chuyến xe ngoại thành: Mã số chuyến, Họ tên tài xế, số xe, nơi đến, số ngày đi được, doanh thu.

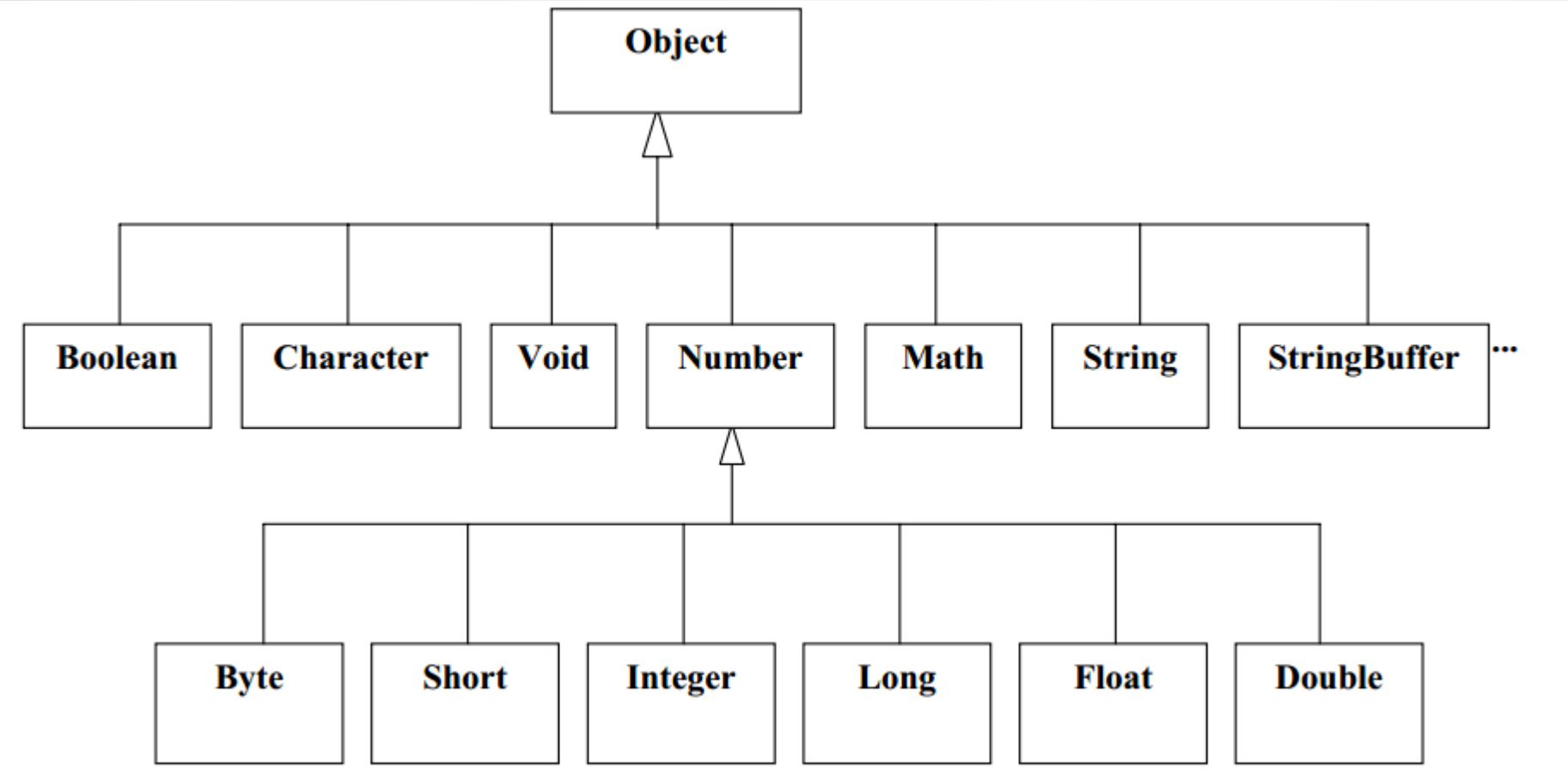
Thực hiện các yêu cầu sau:

- Xây dựng các lớp với chức năng thừa kế.
- Viết chương trình quản lý các chuyến xe theo dạng cây thừa kế với các phương thức sau:
  - Nhập, xuất danh sách các chuyến xe (danh sách có thể dùng cấu trúc mảng).
  - Tính tổng doanh thu cho từng loại xe.

4.6. Case Study

- Xác định
  - Encapsulation: ?
  - Abstraction: ?
  - Inheritance: ? Vẽ mô hình thừa kế
  - Polymorphism: ?
- Xây dựng lớp Chuyến xe bao gồm các thuộc tính chung cho cả chuyến xe ngoại thành và chuyến xe nội thành: *mã chuyến xe, tên tài xế, số xe, doanh thu*.
- Lớp con
  - Xây dựng lớp Chuyến xe Ngoại thành thừa kế lớp Chuyến xe bao gồm thuộc tính: nơi đến, số ngày.
  - Xây dựng lớp Chuyến xe Nội thành thừa kế lớp Chuyến xe bao gồm thuộc tính: số km, số tuyến.

4.7. Một số lớp cơ bản trong Java



4.7. Một số lớp cơ bản trong Java (tt)

Lớp Object

- Mọi lớp trong Java đều được mặc định thừa kế lớp Object dù không khai báo dung từ khóa extends.

  public class SinhVien {...} Tương đương với
  public class SinhVien extends Object {...}

- Một số phương thức trong lớp Object
  - `public String toString()`: trình bày object như là 1 chuỗi
  - `public boolean equals(Object obj)`: dùng để so sánh 2 đối tượng
  - `public int hashCode()`: trả về 1 mã băm dùng trong việc xác định đối tượng trong 1 tập hợp.
  - `Class getClass()`: trả lại tên lớp của đối tượng hiện thời.

4.7. Một số lớp cơ bản trong Java (tt)

Lớp Character

- static boolean isUppercase(char ch)
- static boolean isLowercase(char ch)
- static boolean isDigit(char ch)
- static boolean isLetter(char ch)
- static boolean isLetterOrDigit(char ch)
- static char toUpperCase(char ch)

4.7. Một số lớp cơ bản trong Java (tt)

Lớp String

- Chuỗi ký tự không thay đổi được nội dung
- Khởi tạo
  - String(String),
  - String(StringBuffer)
  - String(byte[]), String(char[])
- Phương thức
  - int length(): kích thước của chuỗi
  - char charAt(int index): ký tự ở vị trí index

4.7. Một số lớp cơ bản trong Java (tt)

Lớp String (tt)

- So sánh chuỗi
  - boolean equals(String)
  - boolean equalsIgnoreCase(String)
  - boolean startWith(String)
  - boolean endWith(String)
  - int compareTo(String)
- Chuyển đổi
  - String toUpperCase()
  - String toLowerCase()
- Nối chuỗi
  - String concat(String)
  - toán tử “+”

4.7. Một số lớp cơ bản trong Java (tt)

Lớp String (tt)

- Tìm kiếm
  - int indexOf(char), int
  - indexOf(char ch, int from)
  - int indexOf(String), int
  - indexOf(String s, int from)
  - int lastIndexOf(char),
  - lastIndexOf(char, int)
  - lastIndexOf(String),
  - lastIndexOf(String, int)

4.7. Một số lớp cơ bản trong Java (tt)

Lớp String (tt)

- Thay thế
  - String replace(char ch, char new_ch)
- Trích chuỗi
  - String trim(): loại bỏ ký tự trắng
  - String substring(int startIndex)
  - String substring(int startIdx, int endIdx)

4.7. Một số lớp cơ bản trong Java (tt)

Lớp StringBuffer

- Chuỗi ký tự thay đổi được nội dung
- Khởi tạo
  - StringBuffer(String)
  - StringBuffer(int length)
  - StringBuffer(): đặt kích thước mặc định 16
- Các phương thức
  - int length(), void setLength()
  - char charAt(int index)
  - void setCharAt(int index, char ch)
  - String toString()

4.7. Một số lớp cơ bản trong Java (tt)

Lớp StringBuffer (tt)

- Thêm, xóa
  - append(String), append(type)
  - insert(int offset, String s),
  - insert(int offset, char[] chs),
  - insert(int offset, type t)
  - delete(int start, int end): xóa chuỗi con
  - delete(int index): xóa một ký tự
  - reverse(): đảo ngược

4.7. Một số lớp cơ bản trong Java (tt)

Lớp StringBuffer (tt)

- Thêm, xóa
  - append(String), append(type)
  - insert(int offset, String s),
  - insert(int offset, char[] chs),
  - insert(int offset, type t)
  - delete(int start, int end): xóa chuỗi con
  - delete(int index): xóa một ký tự
  - reverse(): đảo ngược

4.7. Một số lớp cơ bản trong Java (tt)

Lớp Math

- Hằng số
  - Math.E
  - Math.PI
- Các phương thức static
  - type abs(type)
  - double ceil(double), double floor(double)
  - int round(float), long round(double)
  - type max(type, type), type min(type, type)
  - double random(): sinh số ngẫu nhiên trong đoạn[0.0,1.0]

4.7. Một số lớp cơ bản trong Java (tt)

Lớp Math (tt)

- Lũy thừa
  - double pow(double, double)
  - double exp(double)
  - double log(double)
  - double sqrt(double)
- Lượng giác
  - double sin(double)
  - double cos(double)
  - double tan(double)