Objektum orientált programozás Java-ban

Osztályok definiálása

```
class Car {
  String licensePlate;
  double speed;
  double maxSpeed;
}
```

Objektumok létrehozása (`new`)

```
class Car {
  String licensePlate;
  double speed;
  double maxSpeed;
}
```

```
    Car c;
    c = new Car();
    Car c = new Car();
```

Objektum attribútumainak elérése

```
class Car {
  String licensePlate;
  double speed;
  double maxSpeed;
}
```

```
    Car c = new Car();

    c.licensePlate = "New York A45 636";
    c.speed = 70.0;
    c.maxSpeed = 123.45;

    System.out.println(c.licensePlate + " is moving at " + c.speed +
      "kilometers per hour.");
```

Egy `Car` objektum használata egy másik osztályban

```
class Car {
  String licensePlate;
  double speed;
  double maxSpeed;
}
class CarTest {
  public static void main(String args[]) {
    Car c = new Car();
    c.licensePlate = "New York A45 636";
    c.speed = 70.0;
    c.maxSpeed = 123.45;
    System.out.println(c.licensePlate + " is moving at " + c.speed +
      "kilometers per hour.");
  }
}
```

Attribútumok (tagváltozók)

```
class Car {
  String licensePlate; // tagváltozó
  double speed;        // tagváltozó
  double maxSpeed;     // tagváltozó
}
```

Metódusok (tagfüggvények)

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;        // kilometers per hour
  double maxSpeed;     // kilometers per hour

  // accelerate to maximum speed
  // put the pedal to the metal
  void floorIt()       // metódus
  {
    this.speed = this.maxSpeed;
  }

}
```

A metódusok hívása

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;        // kilometers per hour
  double maxSpeed;     // kilometers per hour

  // accelerate to maximum speed
  // put the pedal to the metal
  void floorIt() {
    this.speed = this.maxSpeed;
  }
}
class CarTest2 {
  public static void main(String args[]) {

    Car c = new Car();

    c.licensePlate = "New York A45 636";
    c.speed = 0.0;
    c.maxSpeed = 123.45;

    System.out.println(c.licensePlate + " is moving at " + c.speed +
     " kilometers per hour.");
    c.floorIt();

    System.out.println(c.licensePlate + " is moving at " + c.speed +
     " kilometers per hour.");

  }

}
```

Tagváltozók használata metódusokban

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;        // kilometers per hour
  double maxSpeed;     // kilometers per hour

  void print() {
    System.out.println(licensePlate + " is moving at " + speed +
     " kilometers per hour ");
  }
}
```

Argumentumok átadása a metódusoknak

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;        // kilometers per hour
```

```
  double maxSpeed;      // kilometers per hour
  void accelerate(double deltaV) {
     speed = speed + deltaV;
     if (speed > maxSpeed) {
       speed = maxSpeed;
     }
     if (speed <  0.0) {
       speed = 0.0;
     }
  }
}
```

A `this` referencia használata - hivatkozás az objektumra

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;         // kilometers per hour
  double maxSpeed;      // kilometers per hour
  void accelerate(double deltaV) {
     speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed <  0.0) {
       this.speed = 0.0;
     }
    }
}
void setNameURLDescription(String s1, String s2, String s3) {
  this.name = s1;
  this.url = s2;
  this.description = s3;
}
```

Példa a paraméterek átadására

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;         // kilometers per hour
  double maxSpeed;      // kilometers per hour

  // accelerate to maximum speed
  // put the pedal to the metal
  void floorIt() {
    this.speed = this.maxSpeed;
  }

  void accelerate(double deltaV) {

     this.speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed <  0.0) {
       this.speed = 0.0;
     }

  }

}
class CarTest3 {

  public static void main(String args[]) {

    Car c = new Car();

    c.licensePlate = "New York A45 636";
    c.speed = 0.0;
    c.maxSpeed = 123.45;
```

```
    System.out.println(c.licensePlate + " is moving at " + c.speed +
     " kilometers per hour.");

    for (int i = 0; i < 15; i++) {
      c.accelerate(10.0);
      System.out.println(c.licensePlate + " is moving at " + c.speed +
       " kilometers per hour.");
    }

  }

}
% java CarTest3
New York A45 636 is moving at 0.0 kilometers per hour.
New York A45 636 is moving at 10.0 kilometers per hour.
New York A45 636 is moving at 20.0 kilometers per hour.
New York A45 636 is moving at 30.0 kilometers per hour.
New York A45 636 is moving at 40.0 kilometers per hour.
New York A45 636 is moving at 50.0 kilometers per hour.
New York A45 636 is moving at 60.0 kilometers per hour.
New York A45 636 is moving at 70.0 kilometers per hour.
New York A45 636 is moving at 80.0 kilometers per hour.
New York A45 636 is moving at 90.0 kilometers per hour.
New York A45 636 is moving at 100.0 kilometers per hour.
New York A45 636 is moving at 110.0 kilometers per hour.
New York A45 636 is moving at 120.0 kilometers per hour.
New York A45 636 is moving at 123.45 kilometers per hour.
New York A45 636 is moving at 123.45 kilometers per hour.
New York A45 636 is moving at 123.45 kilometers per hour.
```

Tagváltozók értékének beállítása metódusok segítségével

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;        // kilometers per hour
  double maxSpeed;     // kilometers per hour

  // setter method for the license plate property
  void setLicensePlate(String licensePlate) {
    this.licensePlate = licensePlate;
  }

  // accelerate to maximum speed
  // put the pedal to the metal
  void floorIt() {
    this.speed = this.maxSpeed;
  }

  void accelerate(double deltaV) {

     this.speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed <  0.0) {
       this.speed = 0.0;
     }

  }

}
```

Metódusok visszatérési értéke

```
  String getLicensePlate() {
    return this.licensePlate;
  }
```

Metódusok visszatérési értéke - példa

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;        // kilometers per hour
  double maxSpeed;     // kilometers per hour

  // getter (accessor) methods
  String getLicensePlate() {
    return this.licensePlate;
  }

  double getMaxSpeed() {
    return this.MaxSpeed;
  }

  double getSpeed() {
    return this.speed;
  }

  // accelerate to maximum speed
  // put the pedal to the metal
  void floorIt() {
    this.speed = this.maxSpeed;
  }

  void accelerate(double deltaV) {
     this.speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed <  0.0) {
       this.speed = 0.0;
     }

  }

}
```

Konstruktorok - I.

```
Car c = new Car();
-------------------------------------------------------------
  Car() {
    this.licensePlate = "";
    this.speed  = 0.0;
    this.maxSpeed = 120.0;
  }

  Car(String licensePlate, double speed, double maxSpeed) {
    this.licensePlate = licensePlate;
    this.speed  = speed;
    this.maxSpeed = maxSpeed;
  }

  Car(String licensePlate, double maxSpeed) {

    this.licensePlate = licensePlate;
    this.speed  = 0.0;
    this.maxSpeed = maxSpeed;
  }
```

Konstruktorok - II.

```
class Car {
  String licensePlate; // e.g. "New York A456 324"
  double speed;        // kilometers per hour
  double maxSpeed;     // kilometers per hour
```

```
  Car(String licensePlate, double maxSpeed) {

    this.licensePlate = licensePlate;
    this.speed  = 0.0;
    this.maxSpeed = maxSpeed;

  }

  // getter (accessor) methods
  String getLicensePlate() {
    return this.licensePlate;
  }

  double getMaxSpeed() {
    return this.speed;
  }

  double getSpeed() {
    return this.maxSpeed;
  }

  // setter method for the license plate property
  void setLicensePlate(String licensePlate) {
    this.licensePlate = licensePlate;
  }

  // accelerate to maximum speed
  // put the pedal to the metal
  void floorIt() {
    this.speed = this.maxSpeed;
  }

  void accelerate(double deltaV) {

     this.speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed <  0.0) {
       this.speed = 0.0;
     }

  }

}
```

A konstruktorok használata

```
class CarTest4 {
  public static void main(String args[]) {

    Car c = new Car("New York A45 636", 123.45);

    System.out.println(c.licensePlate + " is moving at " + c.speed +
     " kilometers per hour.");

    for (int i = 0; i < 15; i++) {
      c.accelerate(10.0);
      System.out.println(c.licensePlate + " is moving at " + c.speed +
       " kilometers per hour.");
    }

  }

}
```

A speed tagváltozó használata az objektumon belül

```
   void accelerate(double deltaV) {
     this.speed = this.speed + deltaV;
```

```
      if (this.speed > this.maxSpeed) {
        this.speed = this.maxSpeed;
      }
      if (this.speed < 0.0) {
        this.speed = 0.0;
      }

  }
  Car(String licensePlate, double maxSpeed) {
    this.licensePlate = licensePlate;
    this.speed  = 0.0;
    if (maxSpeed >= 0.0) {
      this.maxSpeed = maxSpeed;
    }
    else {
      maxSpeed = 0.0;
    }

  }
-------------------------------------------------
  Car c = new Car("New York A234 567", 100.0);
  c.speed = 150.0;
```

Tagváltozók és metódusok láthatóságának (elérhetőségének) szabályozása (access protection)

```
  Car c = new Car("New York A234 567", 100.0);
  c.speed = 150.0;
-----------------------------------------------------------------------
public class Car {
  private String licensePlate; // e.g. "New York A456 324"
  private double speed;        // kilometers per hour
  private double maxSpeed;     // kilometers per hour

  public Car(String licensePlate, double maxSpeed) {
    this.licensePlate = licensePlate;
    this.speed  = 0.0;
    if (maxSpeed >= 0.0) {
      this.maxSpeed = maxSpeed;
    }
    else {
      maxSpeed = 0.0;
    }

  }
  // getter (accessor) methods
  public String getLicensePlate() {
    return this.licensePlate;
  }
  public double getMaxSpeed() {
    return this.speed;
  }
  public double getSpeed() {
    return this.maxSpeed;
  }
  // setter method for the license plate property
  public void setLicensePlate(String licensePlate) {
    this.licensePlate = licensePlate;
  }
  // accelerate to maximum speed
  // put the pedal to the metal
  public void floorIt() {
    this.speed = this.maxSpeed;
  }

  public void accelerate(double deltaV) {
     this.speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
```

```
      }
      if (this.speed <  0.0) {
        this.speed = 0.0;
      }

  }

}
```

Példa: A hozzáférési jogok szintjei - Static tagváltozók használata

```
public class Car {
  private String licensePlate; // e.g. "New York A456 324"
  private double speed;        // kilometers per hour
  private double maxSpeed;     // kilometers per hour
  static String version = "1.0";

  public Car(String licensePlate, double maxSpeed) {
    this.licensePlate = licensePlate;
    this.speed  = 0.0;
    if (maxSpeed >= 0.0) {
      this.maxSpeed = maxSpeed;
    }
    else {
      maxSpeed = 0.0;
    }

  }
  // getter (accessor) methods
  public String getLicensePlate() {
    return this.licensePlate;
  }
  public double getMaxSpeed() {
    return this.speed;
  }
  public double getSpeed() {
    return this.maxSpeed;
  }
  public static String getVersion() {
    return version;
  }
  // setter method for the license plate property
  public void setLicensePlate(String licensePlate) {
    this.licensePlate = licensePlate;
  }
  // accelerate to maximum speed
  // put the pedal to the metal
  public void floorIt() {
    this.speed = this.maxSpeed;
  }

  public void accelerate(double deltaV) {
     this.speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed <  0.0) {
       this.speed = 0.0;
     }

  }

}
```

Hivatkozás a static tagváltozókra

```
  Car c = new Car("New York", 89.7);
  String s = c.getVersion();
  String s = Car.getVersion();
Error:    Can't make static reference to method void print() in
```

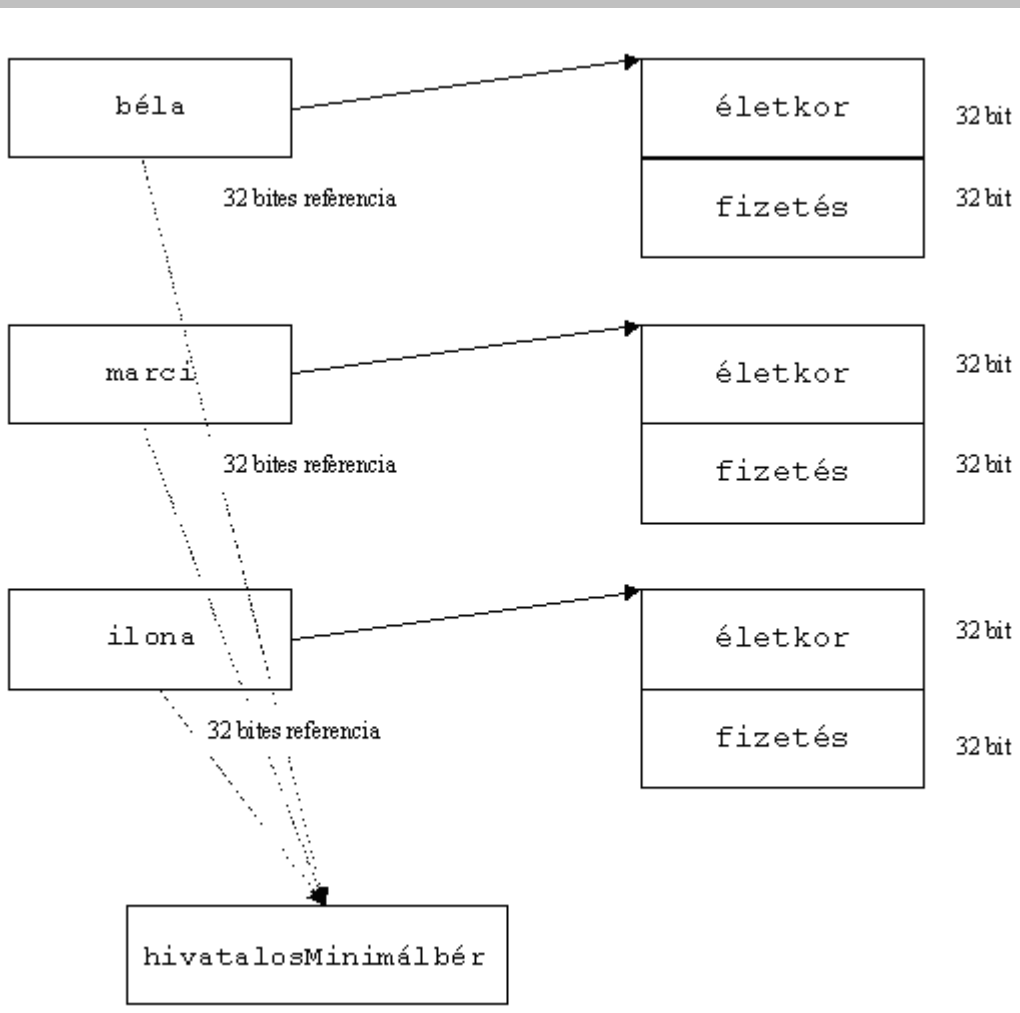
class test.

Static (class) tagváltozók

```
class Alkalmazott {
  int életkor;
  int fizetés;
  static int hivatalosMinimálbér;

  int mennyiFizetésJárNeki()
  {
    if (hivatalosMinimálbér > Math.round(fizetés * (1 + életkor/100.0)))
    {
      return hivatalosMinimálbér;
    }
    else
    {
       return (int) Math.round(fizetés * (1 + életkor/100.0));
    }
  }
}
```

```
Alkalmazott = béla new Alkalmazott();
Alkalmazott = marci new Alkalmazott();
Alkalmazott = ilona new Alkalmazott();
```



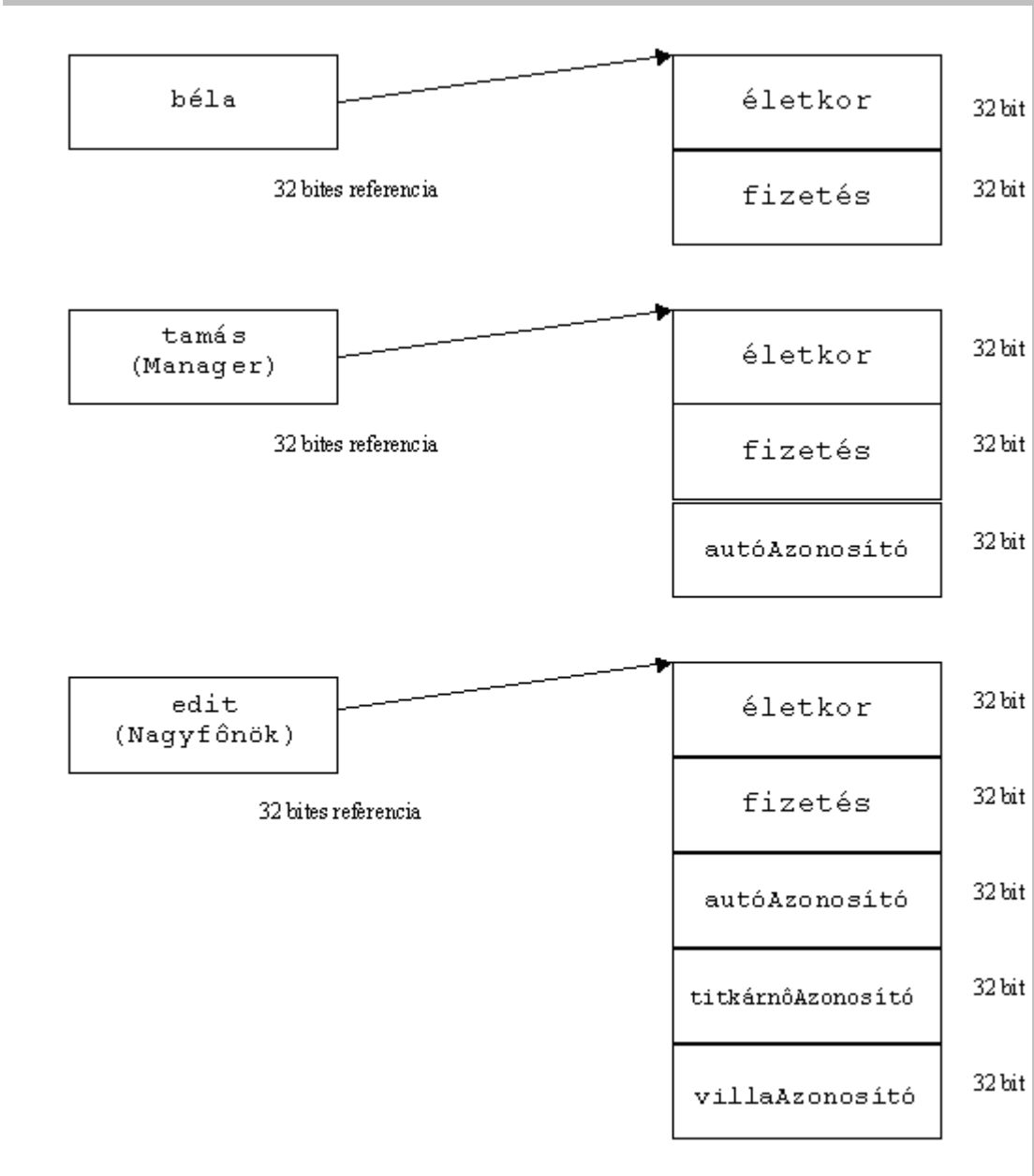
Öröklés

```
class Alkalmazott {
 int életkor;
 int fizetés;
}

class Manager extends Alkalmazott {
 int autóAzonosító;
}

class Nagyfônök extends Manager {
 int titkárnôAzonosító;
 int villaAzonosító;
}


Alkalmazott = béla new Alkalmazott();
Manager = marci new Manager();
Nagyfônök = ilona new Nagyfônök();
```



Overloading

```
public class Car {

  private String licensePlate; // e.g. "New York A456 324"
  private double speed;        // kilometers per hour
  private double maxSpeed;     // kilometers per hour

  // constructors
  public Car(String licensePlate, double maxSpeed) {

    this.licensePlate = licensePlate;
    this.speed  = 0.0;
    if (maxSpeed >= 0.0) {
      this.maxSpeed = maxSpeed;
    }
    else {
      maxSpeed = 0.0;
    }

  }

  public Car(String licensePlate, double speed, double maxSpeed) {

    this.licensePlate = licensePlate;
    if (maxSpeed >= 0.0) {
      this.maxSpeed = maxSpeed;
    }
    else {
      maxSpeed = 0.0;
    }

    if (speed < 0.0) {
      speed = 0.0;
    }

    if (speed <= maxSpeed) {
      this.speed = speed;
    }
    else {
      this.speed = maxSpeed;
    }

  }

  // other methods...

}
Error:    Method Car(double) not found in class Car.
Car.java  line 17
```

`this` használata a konstruktorokban

```
  public Car(String licensePlate, double maxSpeed) {
    Car(String licensePlate, 0.0, double maxSpeed);
  }

  public Car(String licensePlate, double maxSpeed) {
    this(String licensePlate, 0.0, double maxSpeed);
  }

-------------------------------------------------
public class Car {
  private String licensePlate; // e.g. "New York A456 324"
  private double speed;        // kilometers per hour
  private double maxSpeed;     // kilometers per hour
  // constructors
  public Car(String licensePlate, double maxSpeed) {
    this(String licensePlate, 0.0, double maxSpeed);
  }
```

```
  public Car(String licensePlate, double speed, double maxSpeed) {
    this.licensePlate = licensePlate;
    if (maxSpeed >= 0.0) {
      this.maxSpeed = maxSpeed;
    }
    else {
      maxSpeed = 0.0;
    }
    if (speed < 0.0) {
      speed = 0.0;
    }
    if (speed <= maxSpeed) {
      this.speed = speed;
    }
    else {
      this.speed = maxSpeed;
    }
  }
  // other methods...
}
```

Operator Overloading

Inheritance (öröklődés)

```
//---------------------------------------------------------------------------
//  class Car - no inheritance
//---------------------------------------------------------------------------
public class Car {

  private String licensePlate;      // e.g. "New York A456 324"
  private double speed;             // kilometers per hour
  private double maxSpeed;          // kilometers per hour
  private String make;              // e.g. "Ford"
  private String model;             // e.g. "Taurus"
  private int    year;              // e.g. 1997, 1998, 1999, 2000, 2001, etc.
  private int    numberPassengers;  // e.g. 4
  private int    numberWheels = 4;  // all cars have four wheels
  private int    numberDoors;       // e.g. 4

  // constructors
  public Car(String licensePlate, double maxSpeed,
   String make, String model, int year, int numberOfPassengers,
   int numberOfDoors) {
    this(licensePlate, 0.0, maxSpeed, make, model, year, numberOfPassengers,
     numberOfDoors);
  }
  public Car(String licensePlate, double speed, double maxSpeed,
   String make, String model, int year, int numberOfPassengers) {
    this(licensePlate, speed, maxSpeed, make, model, year,
     numberOfPassengers, 4);
  }
  public Car(String licensePlate, double speed, double maxSpeed,
   String make, String model, int year, int numberOfPassengers,
   int numberOfDoors) {
    // I could add some more constraints like the
    // number of doors being positive but I won't
    // so that this example doesn't get too big.
    this.licensePlate = licensePlate;
    this.make = make;
    this.model = model;
    this.year = year;
    this.numberPassengers = numberOfPassengers;
    this.numberDoors = numberOfDoors;
    if (maxSpeed >= 0.0) {
      this.maxSpeed = maxSpeed;
    }
    else {
      maxSpeed = 0.0;
    }
    if (speed < 0.0) {
```

```
      speed = 0.0;
    }
    if (speed <= maxSpeed) {
      this.speed = speed;
    }
    else {
      this.speed = maxSpeed;
    }
  }

  // getter (accessor) methods
  public String getLicensePlate() {
    return this.licensePlate;
  }
  public String getMake() {
    return this.make;
  }
  public String getModel() {
    return this.model;
  }
  public int getYear() {
    return this.year;
  }
  public int getNumberOfPassengers() {
    return this.numberPassengers;
  }
  public int getNumberOfWheels() {
    return this.numberWheels;
  }
//  public int getNumberOfDoors() {
//    return this.numberDoors;
//  }
  public double getMaxSpeed() {
    return this.speed;
  }
  public double getSpeed() {
    return this.maxSpeed;
  }
  // setter method for the license plate property
  public void setLicensePlate(String licensePlate) {
    this.licensePlate = licensePlate;
  }
  // accelerate to maximum speed
  // put the pedal to the metal
  public void floorIt() {
    this.speed = this.maxSpeed;
  }
  public void accelerate(double deltaV) {
     this.speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed <  0.0) {
       this.speed = 0.0;
     }
  }
}

//--------------------------------------------------------------------------
//  class Motorcycle - no inheritance
//--------------------------------------------------------------------------
public class Motorcycle {

  private String licensePlate;      // e.g. "New York A456 324"
  private double speed;             // kilometers per hour
  private double maxSpeed;          // kilometers per hour
  private String make;              // e.g. "Harley-Davidson"
  private String model;             // e.g. "panhead"
  private int    year;              // e.g. 1997, 1998, 1999, 2000, 2001, etc.
  private int    numberPassengers;  // e.g. 4
  private int    numberWheels = 2;  // all motorcycles have two wheels

  // constructors
  public Motorcycle(String licensePlate, double maxSpeed,
```

```
 String make, String model, int year, int numberOfPassengers) {
  this(licensePlate, maxSpeed, make, model, year, numberOfPassengers);
}
public Motorcycle(String licensePlate, double speed, double maxSpeed,
 String make, String model, int year, int numberOfPassengers) {
  this(licensePlate, speed, maxSpeed, make, model, year,
   numberOfPassengers, 2);
}
public Motorcycle(String licensePlate, double speed, double maxSpeed,
 String make, String model, int year, int numberOfPassengers, int numberOfWheels) {
  // I could add some more constraints like the
  // number of doors being positive but I won't
  // so that this example doesn't get too big.
  this.licensePlate = licensePlate;
  this.make = make;
  this.model = model;
  this.year = year;
  this.numberPassengers = numberOfPassengers;
  if (maxSpeed >= 0.0) {
    this.maxSpeed = maxSpeed;
  }
  else {
    maxSpeed = 0.0;
  }
  if (speed < 0.0) {
    speed = 0.0;
  }
  if (speed <= maxSpeed) {
    this.speed = speed;
  }
  else {
    this.speed = maxSpeed;
  }
}

// getter (accessor) methods
public String getLicensePlate() {
  return this.licensePlate;
}
public String getMake() {
  return this.make;
}
public String getModel() {
  return this.model;
}
public int getYear() {
  return this.year;
}
public int getNumberOfPassengers() {
  return this.numberPassengers;
}
public int getNumberOfWheels() {
  return this.numberWheels;
}
public int getNumberOfDoors() {
  return this.numberDoors;
}
public double getMaxSpeed() {
  return this.speed;
}
public double getSpeed() {
  return this.maxSpeed;
}
// setter method for the license plate property
public void setLicensePlate(String licensePlate) {
  this.licensePlate = licensePlate;
}
// accelerate to maximum speed
// put the pedal to the metal
public void floorIt() {
  this.speed = this.maxSpeed;
}
public void accelerate(double deltaV) {
   this.speed = this.speed + deltaV;
```

```
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed < 0.0) {
       this.speed = 0.0;
     }
  }
}


//---------------------------------------------------------------------------
//  class MotorVehicle - common parent
//---------------------------------------------------------------------------
public class MotorVehicle {
  protected String licensePlate;      // e.g. "New York A456 324"
  protected double speed;             // kilometers per hour
  protected double maxSpeed;          // kilometers per hour
  protected String make;              // e.g. "Harley-Davidson", "Ford"
  protected String model;             // e.g. "Fatboy", "Taurus"
  protected int    year;              // e.g. 1998, 1999, 2000, 2001, etc.
  protected int    numberPassengers;  // e.g. 4
  protected int    numberWheels;

  // constructors
  public MotorVehicle(String licensePlate, double maxSpeed,
   String make, String model, int year, int numberOfPassengers) {
    this(licensePlate, 0.0 ,maxSpeed, make, model, year, numberOfPassengers);
  }

//  public MotorVehicle(String licensePlate, double speed, double maxSpeed,
//   String make, String model, int year, int numberOfPassengers) {
//    this(licensePlate, speed, maxSpeed, make, model, year, numberOfPassengers);
//  }

  public MotorVehicle(String licensePlate, double speed, double maxSpeed,
   String make, String model, int year, int numberOfPassengers) {
    // I could add some more constraints like the
    // number of doors being positive but I won't
    // so that this example doesn't get too big.
    this.licensePlate = licensePlate;
//    this.make = make;
    this.model = model;
    this.year = year;
    this.numberPassengers = numberOfPassengers;
    if (maxSpeed >= 0.0) {
      this.maxSpeed = maxSpeed;
    }
    else {
      maxSpeed = 0.0;
    }
    if (speed < 0.0) {
      speed = 0.0;
    }
    if (speed <= maxSpeed) {
      this.speed = speed;
    }
    else {
      this.speed = maxSpeed;
    }
  }

  // getter (accessor) methods
  public String getLicensePlate() {
    return this.licensePlate;
  }
  public String getMake() {
    return this.make;
  }
  public String getModel() {
    return this.model;
  }
  public int getYear() {
    return this.year;
  }
```

```
  public int getNumberOfPassengers() {
    return this.numberPassengers;
  }
  public int getNumberOfWheels() {
    return this.numberWheels;
  }
  public double getMaxSpeed() {
    return this.speed;
  }
  public double getSpeed() {
    return this.maxSpeed;
  }

  // setter method for the license plate property
  protected void setLicensePlate(String licensePlate) {
    this.licensePlate = licensePlate;
  }
  // accelerate to maximum speed
  // put the pedal to the metal
  public void floorIt() {
    this.speed = this.maxSpeed;
  }
  public void accelerate(double deltaV) {
     this.speed = this.speed + deltaV;
     if (this.speed > this.maxSpeed) {
       this.speed = this.maxSpeed;
     }
     if (this.speed < 0.0) {
       this.speed = 0.0;
     }
  }
}

//-------------------------------------------------------------------------
//  class  Motorcycle - inheritance
//-------------------------------------------------------------------------

public class Motorcycle extends MotorVehicle {

  protected int numberWheels = 2;

  // constructors
  public Motorcycle(String licensePlate, double maxSpeed,
   String make, String model, int year, int numberOfPassengers) {
    this(licensePlate, 0.0, maxSpeed, make, model, year, numberOfPassengers);
  }

  public Motorcycle(String licensePlate, double speed, double maxSpeed,
   String make, String model, int year, int numberOfPassengers) {

    // invoke superclass constructor
    super(licensePlate, speed, maxSpeed, make, model, year,
     numberOfPassengers);
  }
  public int getNumberOfWheels() {
    return this.numberWheels;
  }
}

//-------------------------------------------------------------------------
//  class Car - inheritance
//-------------------------------------------------------------------------
public class Car extends MotorVehicle {

  protected int numberWheels = 4;
  protected int numberDoors;

  // constructors
  public Car(String licensePlate, double maxSpeed,
   String make, String model, int year, int numberOfPassengers,
   int numberOfDoors) {
    this(licensePlate, 0.0, maxSpeed, make, model, year, numberOfPassengers,
     numberOfDoors);
  }
```

```
  public Car(String licensePlate, double speed, double maxSpeed,
   String make, String model, int year, int numberOfPassengers) {
    this(licensePlate, speed, maxSpeed, make, model, year,
     numberOfPassengers, 4);
  }
  public Car(String licensePlate, double speed, double maxSpeed,
   String make, String model, int year, int numberOfPassengers,
   int numberOfDoors) {
    super(licensePlate, speed, maxSpeed, make, model, year, numberOfPassengers);
    this.numberDoors = numberOfDoors;
  }
  public int getNumberOfWheels() {
    return this.numberWheels;
  }
  public int getNumberOfDoors() {
    return this.numberDoors;
  }
}
```

Subosztályok és polimorfizmus

Többszintű öröklődés

```
//---------------------------------------------------------------------------
//  class SlowCar - inheritance
//---------------------------------------------------------------------------
public class SlowCar extends Car {

  private double speedLimit = 112.65408; // kph == 70 mph

  public SlowCar(String licensePlate, double speed, double maxSpeed,
   String make, String model, int year, int numberOfPassengers, int numDoors)
   {
    super(licensePlate, speed, maxSpeed, make, model, year, numberOfPassengers,
numDoors);
     if (speed > speedLimit)
     {
        speed  = speedLimit;
     }
  }
  public void accelerate(double deltaV) {
     double speed = this.speed + deltaV;
     if (speed > this.maxSpeed) {
       speed = this.maxSpeed;
     }
     if (speed > speedLimit) {
       speed = speedLimit;
     }
     if (speed < 0.0) {
       speed = 0.0;
     }
     this.speed = speed;
  }
}
```

A Java osztálykönyvtárak (class library), csomagok

- package java.applet
- package java.awt
- package java.awt.datatransfer
- package java.awt.event
- package java.awt.image
- package java.awt.peer
- package java.beans
- package java.io
- package java.lang

- package java.lang.reflect
- package java.math
- package java.net
- package java.rmi
- package java.rmi.dgc
- package java.rmi.registry
- package java.rmi.server
- package java.security
- package java.security.acl
- package java.security.interfaces
- package java.sql
- package java.text
- package java.util
- package java.util.zip

Példa: a java.net csomag

A java.net interfészei

- ContentHandlerFactory
- FileNameMap
- SocketImplFactory
- URLStreamHandlerFactory

A java.net osztályai

- ContentHandler
- DatagramPacket
- DatagramSocket
- DatagramSocketImpl
- HttpURLConnection
- InetAddress
- MulticastSocket
- ServerSocket
- Socket
- SocketImpl
- URL
- URLConnection
- URLEncoder
- URLStreamHandler

A java.net különleges állapotai

- BindException
- ConnectException
- MalformedURLException
- NoRouteToHostException
- ProtocolException
- SocketException
- UnknownHostException
- UnknownServiceException

Az URL osztály az osztálykönyvtárban

```
public URL(String protocol, String host, int port, String file)
 throws MalformedURLException
```

```
public URL(String protocol, String host, String file) throws MalformedURLException
public URL(String spec) throws MalformedURLException
public URL(URL context, String spec) throws MalformedURLException
public int getPort()
public String getFile()
public String getProtocol()
public String getHost()
public String getRef()
public boolean equals(Object obj)
public int hashCode()
public boolean sameFile(URL other)
public String toString()
public URLConnection openConnection() throws IOException
public final InputStream openStream() throws IOException
public static synchronized void
 setURLStreamHandlerFactory(URLStreamHandlerFactory factory)
```

Egy osztály használata az osztálykönyvtárból

```
public class URLSplitter {

  public static void main(String[] args) {

    for (int i = 0; i < args.length; i++) {
      try {
        java.net.URL u = new java.net.URL(args[i]);
        System.out.println("Protocol: " + u.getProtocol());
        System.out.println("Host: " + u.getHost());
        System.out.println("Port: " + u.getPort());
        System.out.println("File: " + u.getFile());
        System.out.println("Ref: " + u.getRef());
      }
      catch (java.net.MalformedURLException e) {
        System.err.println(args[i] + " is not a valid URL");
      }
    }

  }
}

% java SplitURL http://www.poly.edu
Protocol: http
Host: www.poly.edu
Port: -1
File: /
Ref: null
```

Osztályok importálása

```
import java.net.URL;
import java.net.MalformedURLException;

public class URLSplitter {

  public static void main(String[] args) {

    for (int i = 0; i < args.length; i++) {
      try {
        URL u = new URL(args[i]);
        System.out.println("Protocol: " + u.getProtocol());
        System.out.println("Host: " + u.getHost());
        System.out.println("Port: " + u.getPort());
        System.out.println("File: " + u.getFile());
        System.out.println("Ref: " + u.getRef());
      }
      catch (MalformedURLException e) {
        System.err.println(args[i] + " is not a valid URL");
      }
    }
```

```
  }
}
```

Csomagok importálása

```
import java.net.*;

public class URLSplitter {

  public static void main(String[] args) {

    for (int i = 0; i < args.length; i++) {
      try {
        URL u = new URL(args[i]);
        System.out.println("Protocol: " + u.getProtocol());
        System.out.println("Host: " + u.getHost());
        System.out.println("Port: " + u.getPort());
        System.out.println("File: " + u.getFile());
        System.out.println("Ref: " + u.getRef());
      }
      catch (MalformedURLException e) {
        System.err.println(args[i] + " is not a valid URL");
      }
    }

  }
}
```

A `final` kulcsszó

final osztályok

```
public final class String
```

final metódusok

```
public final String convertCurrency()
```

final mezők

```
public class Physics {
  public static final double c = 2.998E8;
}
```

final argumentumok

abstract

```
public abstract class MotorVehicle {}
```

Interfészek

Az interfészek jelentik az absztrakció következő szintjét.

```
public interface Import {
  public double calculateTariff();
}
```

Az interfészek implementálása

```
public class Car extends MotorVehicle implements Import {

  int numWheels = 4;

  public double calculateTariff() {
    return this.price * 0.1;
  }

}
```

```
import java.io.*;

public class Car extends MotorVehicle
 implements Import, Serializable, Cloneable {

  int numWheels = 4;

  public double calculateTariff() {
    return this.price * 0.1;
  }

}
```

Saját csomagok használata

```
package com.macfaq.net;
import java.net.*;

public class URLSplitter {
  public static void main(String[] args) {
    for (int i = 0; i < args.length; i++) {
      try {
        URL u = new URL(args[i]);
        System.out.println("Protocol: " + u.getProtocol());
        System.out.println("Host: " + u.getHost());
        System.out.println("Port: " + u.getPort());
        System.out.println("File: " + u.getFile());
        System.out.println("Ref: " + u.getRef());
      }
      catch (MalformedURLException e) {
        System.err.println(args[i] + " is not a valid URL");
      }
    }
  }
}
```

```
% javac -d /home/elharo/classes SplitURL.java
```

Kivételek kezelése (Exceptions)

try-catch

```
public class HelloThere {

  public static void main(String[] args) {

    try {
      System.out.println("Hello " + args[0]);
    }
    catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Hello Whoever you are.");
    }
  }
}
```

A `finally` kulcsszó

```
public class HelloThere {

  public static void main(String[] args) {

    try {
      System.out.println("Hello " + args[0]);
    }
    catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Hello Whoever you are.");
    }
    finally {
      System.out.println("How are you?");
    }
  }
}
```