ООП

Інкапсуляція

```
public class Car {
    public static int count;
    public int id;
    public String _maker;
    public double _price;
    public String _year;
    public String _color;

    //конструктор без параметрів
    public Car() {
        count++;
        id = count;    }

    //конструктор з параметрами, який ініціалізує всі поля класу
    public Car(String maker, String color, double price, String year) {
        _maker = maker;
        _price = price;
        _year = year;
        _color = color;
        count++;
        id = count;    }

    //заміщення (перевизначення) методу toString() класу Object
    @Override
    public String toString() {
        return "Авто " + id + " " + _maker + " " + _color + " " + _price + " " + _year + " ";    }
```

Успадкування

Батьківський клас

```
public class SimpleRoom {

   protected double width=0.0;
   protected double length=0.0;

   public SimpleRoom(double width, double
length) {
      this.width=width;
      this.length=length;
      System.out.println("SimpleRoom
створено");    }

   public void info (){
      System.out.println("Кімната: ширина
= "+width+", довжина = "+length);
      System.out.println("Площа кімнати:
"+width*length);     }

   public static void main(String[] args) {
      SimpleRoom s=new SimpleRoom(5, 5);
      s.info();
   } }
```

Дочірній клас

```
public class SimpleRoom2 extends SimpleRoom {
   protected double height;
   public SimpleRoom2(double w, double l, double h) {
      super(w, l);
      height=h;
      System.out.println("SimpleRoom2 створено"); }

   public void info2(){
      System.out.println("Кімната: ширина =
"+super.width+", довжина = "+super.length+", висота=
"+this.height);
      System.out.println("Площа кімнати:
"+width*length); // якщо немає конфлікту з
іменами, то можна і пропустити super
      System.out.println("Об'єм кімнати:
"+width*length*height);
   }

   public static void main(String[] args) {
      SimpleRoom2 s2 = new SimpleRoom2(5, 5, 3);
      System.out.println("Метод info SimpleRoom");
      s2.info();
      System.out.println();
      System.out.println("Метод info2 SimpleRoom2");
      s2.info2();
   } }
```

Поліморфізм

```
Soldier s = new Soldier("Солдат");  // звичайне створення об'єкту Soldier
Soldier s2 = new General("Генерал");  // об'єктна змінна типу Soldier посилається на об'єкт типу General
```

```
General g=new Soldier("Солдат");    // !!! Помилка приведення типу (солдат не генерал)
```

```
General g=(General)new Soldier("Солдат");   // ПОМИЛКА ВИКОНАННЯ!!!
```

```
Soldier sg= new General("Генерал");  //змінна sg посилається на об'єкт типу General
sg.getHealth();                  //методи класу Soldier доступні
//   sg.getSlogan();              //методи класу General недоступні
```

```
General general=(General)sg;    // наш Генерал тепер повноцінний
general.getSlogan();           // метод класу General доступний
```

Абстрактні класи

Абстрактний клас – клас у якоме не реалізовані всі

```
public abstract class CarCost {

   /** обчислення собівартості
    * @return - Собівартість автомобіля на заводі
    */
    public double countPrimeCost() {
       //обчислення собівартості
       return 50000.0;      }

   /** Обчислення вартості перевезення
    * @param Country - країна
    * @return - вартість перевезення
    */
   public abstract double countTransportationCosts(String Country);
   /** Обчислення вартості автомобіля в салонах продажів
    * @return - остаточна ціна автомобіля у певній країні
    */
   public abstract double countLocaleCost();
}
```

Інтерфейси

Інтерфейси (interfaces) – посилальний тип даних, що може містити лише константи, сигнатуру методу та вкладені типи.

Інтерфейс з використанням узагальнень

```
public interface MyInterface <T> {
    public String MYCONST="It is constant"; // константа
    public int method(T o); // нереалізований метод
}
```

Інтерфейс без використання узагальнень

```
public interface MyInterface {
    public String MYCONST="It is constant" // константа
    public int method1(Object o); // нереалізований метод1
    public int method2(String str); // нереалізований метод2
}
```