Методи. ООП

Методи

Метод в Java – закінчена послідовність інструкцій, спрямованих на вирішення окремого завдання

Визначення методу

```
тип-повернення ідентифікатор-методу (параметри)
{
   тіло методу
      return повертаєме-значення // якщо void, то
інструкція return не обов'язкова
}
```

Приклад використання

```
public class ProgramSum{
  public static void main(String[] args) {
    int a=2, b=3, k=4;
    int sum=sum(a, b, k);
    System.out.println("Сума трьох чисел дорівнює "+sum);
  }
  public static int sum (int a, int b, int c){
    return a+b+c;
  }
}
```

Перевантаження методів

```
public class ProgramSum{
 public static void main(String[] args) {
    int a=2, b=3, k=4;
    double a1=2.10, b1=4.20, k1=5.30;
    int sum=sum(a, b, k);
    double sum2=sum(a1, b1, k1);
     System.out.println("Сума трьох цілих чисел дорівнює:
"+sum);
         System.out.println("Сума  трьох  дробових  чисел
дорівнює: "+sum2);
  }
  public static int sum (int a, int b, int c){
    return a+b+c;
  }
  public static double sum (double a, double b, double c){
    return a+b+c;
  }
}
```

Заміщення методів

Клас Parent:

```
public class Parent {

   public static void helloStatic(){
      System.out.println("Hello from Parent.helloStatic()");
   }
   public void helloNonStatic(){
      System.out.println("Hello from Parent.helloNonStatic()");
   }
   public void hello(){
      System.out.println("Hello from Parent.hello()");
   }
}
```

Заміщення методів

```
public class Child extends Parent {
  public static void helloStatic(){
    System.out.println("Hello from Child.helloStatic()");
  }
  @Override
  public void helloNonStatic(){
    System.out.println("Hello from Child.helloNonStatic()");
  }
  public static void main(String[] args) {
    Child child=new Child(); //створюємо об'єкт Child
    child.hello();//викликаємо успадкований метод
    child.helloNonStatic();//викликаємо заміщений метод
    child.helloStatic();//статичні методи рекомендують
викликати через клас, а не через об'єкт, тобто краще писати
Child.helloStatic(). далі використовується поліморфізм -
використовуємо об'єктну змінну батьківського класу і
присвоюємо їй посилання дочірнього класу
    Parent parent=new Child();
    parent.helloNonStatic(); //заміщення спрацьовує
    parent.helloStatic(); //заміщення не спрацьовує,
викликається метод батьківського класу
  }
```

Основні поняття ООП

- **Клас** - певна абстрактна сутність, що на програмному рівні представлена змінними (полями) та методами, що оперують над цими полями.
- **Об'єкт** - конкретний екземпляр класу.
- **Успадкування** - утворення нових класів на основі інших.
- **Інтерфейс** - посилальний тип даних. Інтерфейси схожі на класи, проте їхні поля даних є константами, а методи не реалізовані.
- **Пакети** - каталоги, у яких розміщуються класи.

Основні принципи ООП

- **інкапсуляція** (incapsulation) - концепція побудови класів через закриття їхньої реалізації.
- **успадкування** (inheritance) - створення одних класів на основі інших.
- **поліморфізм** (polymorphism) - можливість використання батьківських класів замість класів нащадків.

Об’єкти та об’єктні змінні

В об’єктах виділяють:

- **Поведінку об’єкту** – що можна з робити з даним об’єктом, або які методи можна застосовувати до нього
- **Стан об’єкту** – те як об’єкт змінюється, коли Ви застосовуєте його методи
- **Ідентичність об’єкту** – відмінність об’єкту від інших об’єктів. Об’єкти можуть мати однаковий стан, проте все рівно вони ідентифікуються як різні об’єкти.

Створення нового об’єкту

new Date(); // створюємо об’єкт, який містить поточний час

Створення об’єктної змінної

Date curDate = new Date();

Пакети

- **З указанням повної назви пакету перед використовуваним класом:**

```
java.util.Date today = new java.util.Date();
```

- **Імпорт класу**

```
import java.util.Date; // імпортуємо класс
```

- **Імпорт пакету**

```
import java.util.*;
```

Керування доступом в пакеті

	Private	Без модифікатора	Protected	Public
Той же клас	Так	Так	Так	Так
Підклас класу в цьому ж пакеті	Ні	Так	Так	Так
Клас цього ж пакету, що не є підкласом	Ні	Так	Так	Так
Підклас класу в іншому пакеті	Ні	Ні	Так	Так
Клас в другому пакеті, що не є підкласом класу даного пакету	Ні	Ні	Ні	Так