НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Паттерны (шаблоны) проектирования

## Порождающие паттерны

Евгений Беркунский
http://www.berkut.mk.ua
eberkunsky@gmail.com

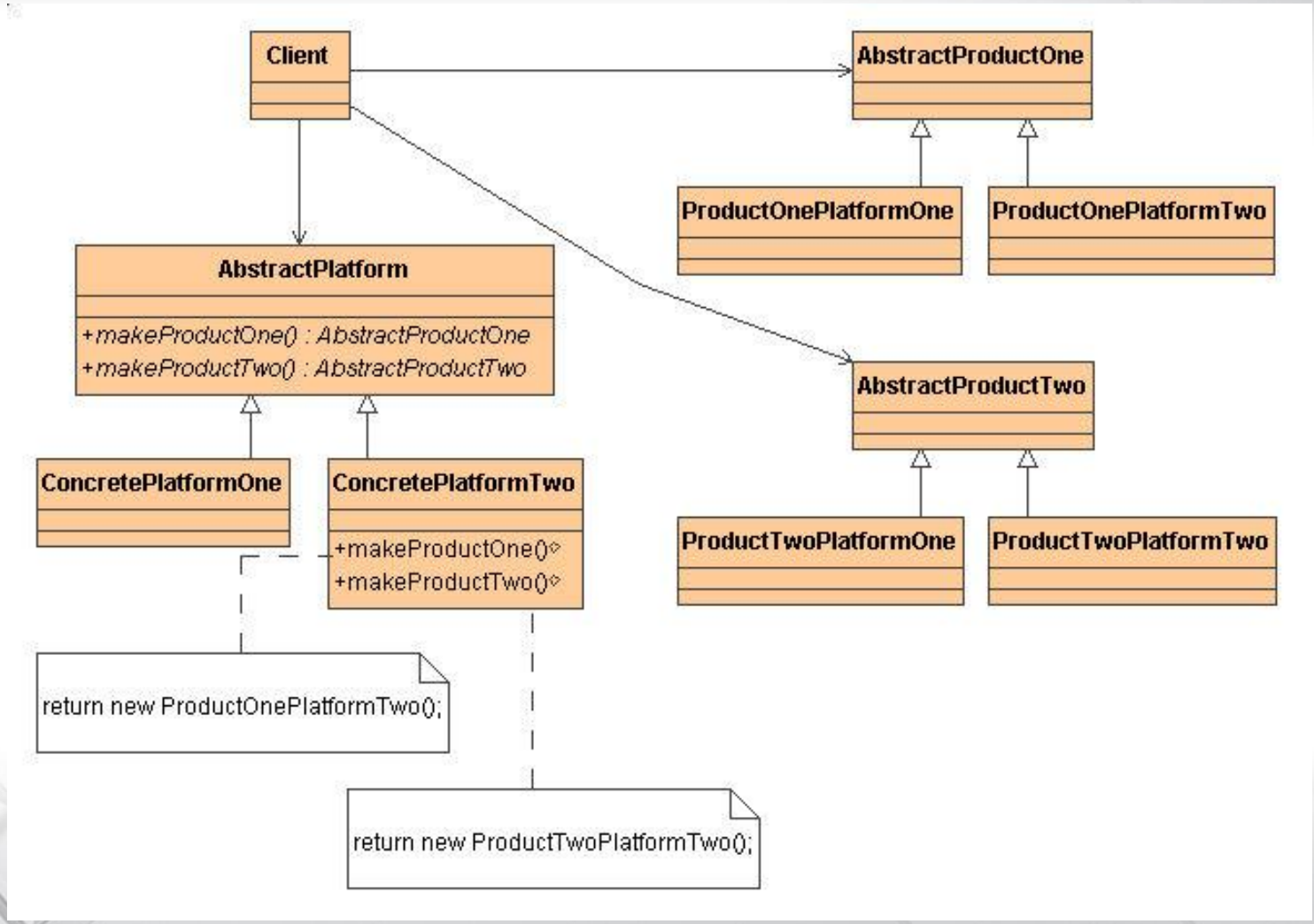
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Порождающие паттерны

- Абстрактная фабрика (abstract factory)
- Строитель (builder)
- Фабричный метод (factory method)
- Прототип (prototype)
- Одиночка (singleton)
- Объектный пул (object pool)
- Ленивая инициализация (lazy initialization)

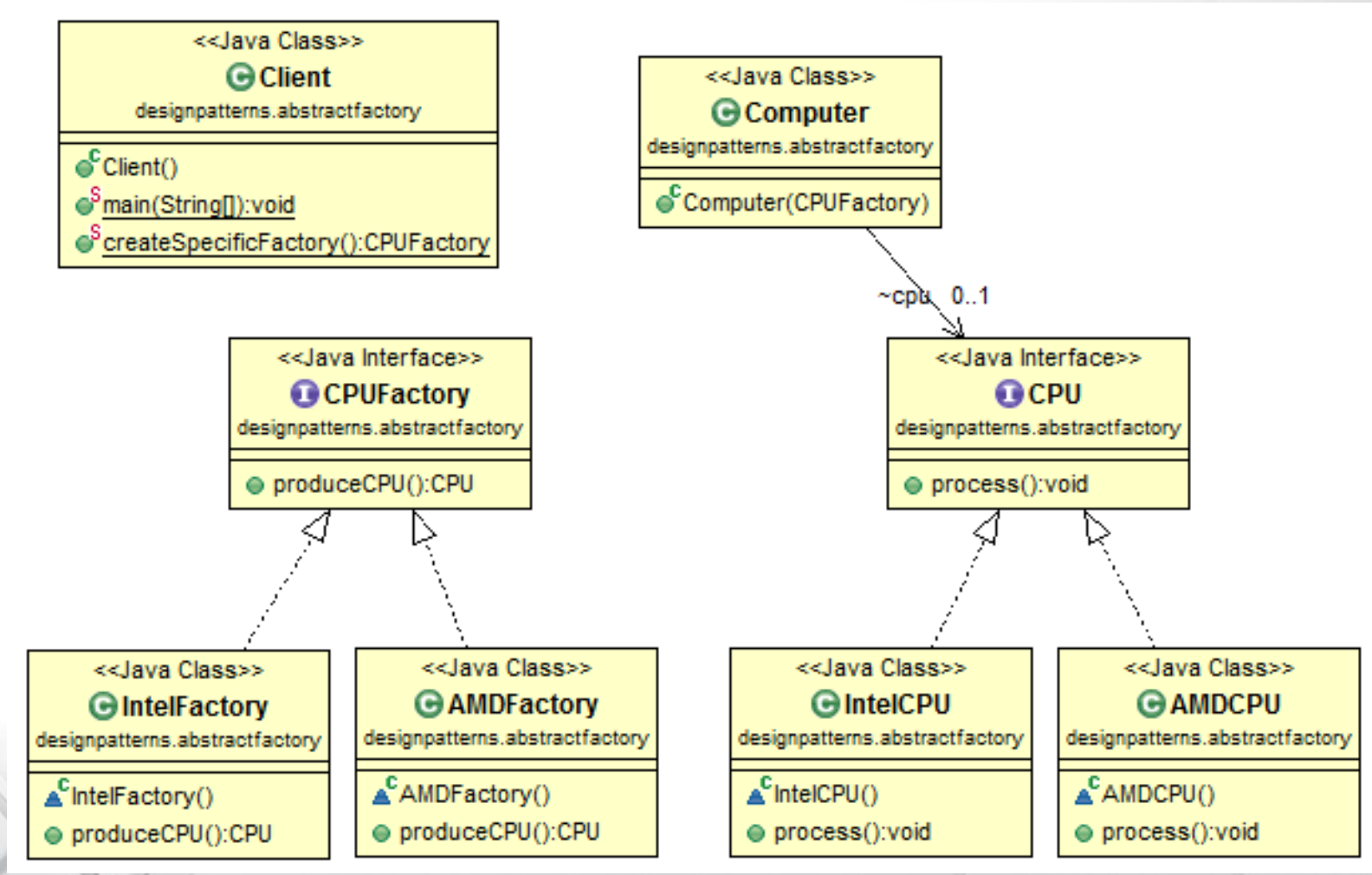
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Абстрактная фабрика
# Abstract Factory



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Абстрактная фабрика
# Abstract factory



# Абстрактная фабрика
# Abstract factory

```
interface CPU {
    void process();
}
interface CPUFactory {
    CPU produceCPU();
}
class AMDFactory implements CPUFactory {
    public CPU produceCPU() {
        return new AMDCPU();
    }
}
class IntelFactory implements CPUFactory {
    public CPU produceCPU() {
        return new IntelCPU();
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Абстрактная фабрика
# Abstract factory

```
class AMDCPU implements CPU {
    public void process() {
        System.out.println("AMD is processing...");
    }
}
class IntelCPU implements CPU {
    public void process() {
        System.out.println("Intel is processing...");
    }
}
class Computer {
    CPU cpu;
    public Computer(CPUFactory factory) {
        cpu = factory.produceCPU();
        cpu.process();
    }
}
```

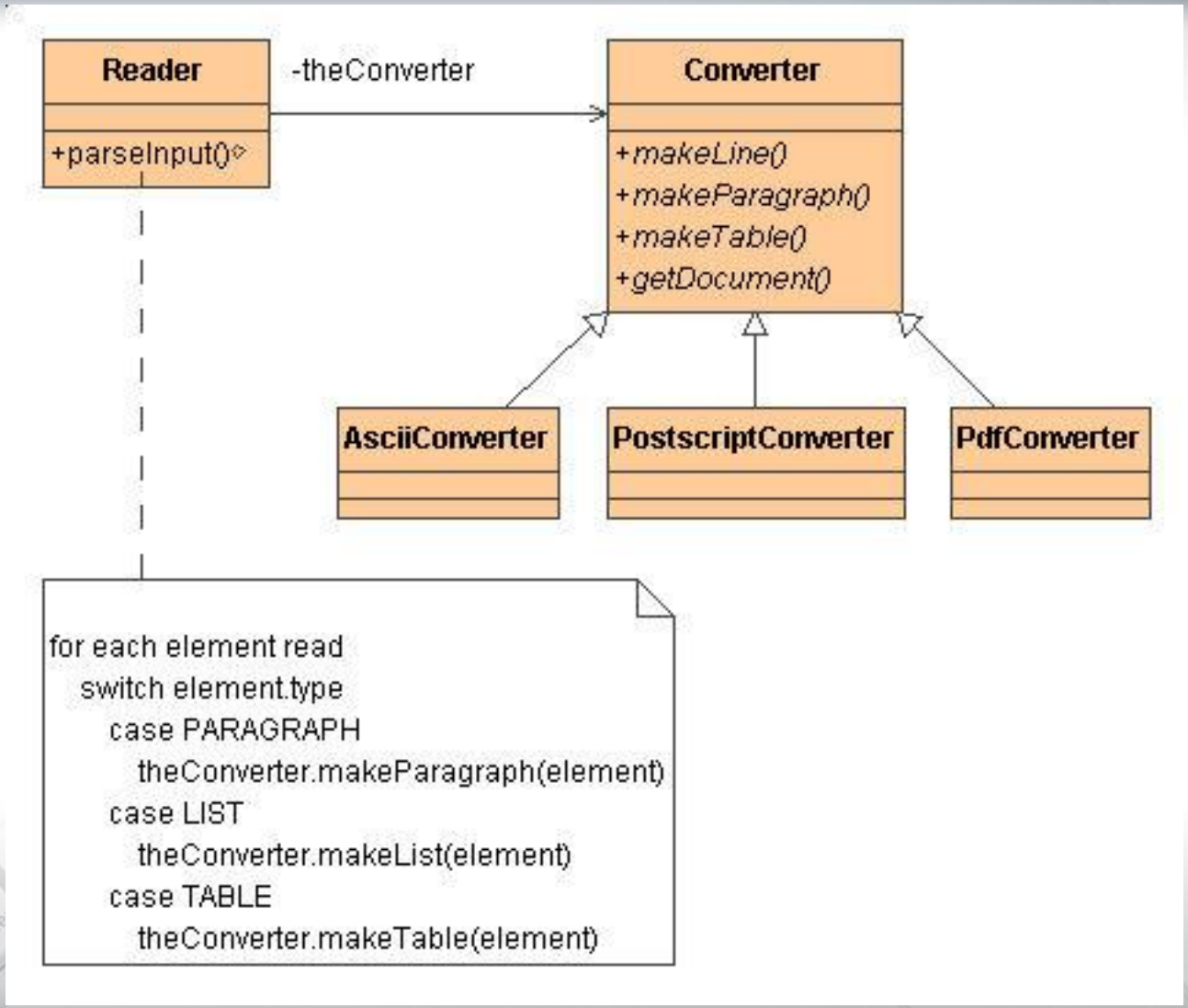
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Абстрактная фабрика
# Abstract factory

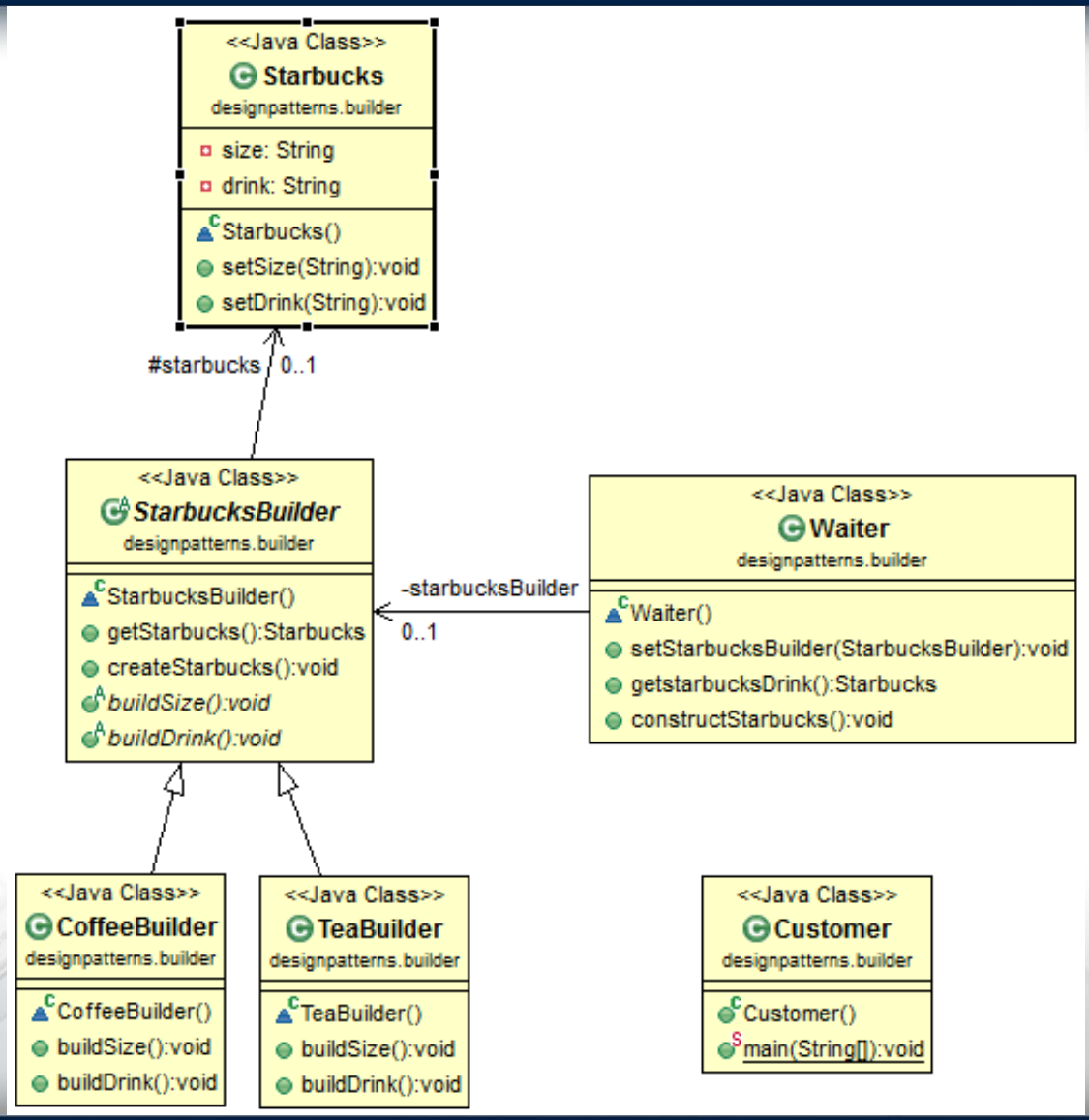
```
public class Client {
    public static void main(String[] args) {
        new Computer(createSpecificFactory());
    }
    public static CPUFactory createSpecificFactory() {
        int sys = 0; // based on specific requirement
        if (sys == 0)
            return new AMDFactory();
        else return new IntelFactory();
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Строитель / Builder



# Строитель / Builder



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Строитель / Builder

```
class Starbucks {
    private String size;
    private String drink;
    public void setSize(String size) {
        this.size = size;
    }
    public void setDrink(String drink) {
        this.drink = drink;
    }
}
//abstract builder
abstract class StarbucksBuilder {
    protected Starbucks starbucks;
    public Starbucks getStarbucks() {
        return starbucks;
    }
    public void createStarbucks() {
        starbucks = new Starbucks();
        System.out.println("a drink is created");
    }
    public abstract void buildSize();
    public abstract void buildDrink();
}
```

# Строитель / Builder

```
// Concrete Builder to build tea
class TeaBuilder extends StarbucksBuilder {
    public void buildSize() {
        starbucks.setSize("large");
        System.out.println("build large size");
    }
    public void buildDrink() {
        starbucks.setDrink("tea");
        System.out.println("build tea");
    }
}
// Concrete builder to build coffee
class CoffeeBuilder extends StarbucksBuilder {
    public void buildSize() {
        starbucks.setSize("medium");
        System.out.println("build medium size");
    }
    public void buildDrink() {
        starbucks.setDrink("coffee");
        System.out.println("build coffee");
    }
}
```

# Строитель / Builder

```
//director to encapsulate the builder
class Waiter {
    private StarbucksBuilder starbucksBuilder;
    public void setStarbucksBuilder(StarbucksBuilder builder){
        starbucksBuilder = builder;
    }
    public Starbucks getstarbucksDrink() {
        return starbucksBuilder.getStarbucks();
    }
    public void constructStarbucks() {
        starbucksBuilder.createStarbucks();
        starbucksBuilder.buildDrink();
        starbucksBuilder.buildSize();
    }
}
//customer
public class Customer {
    public static void main(String[] args) {
        Waiter waiter = new Waiter();
        StarbucksBuilder coffeeBuilder = new  CoffeeBuilder();
        waiter.setStarbucksBuilder(coffeeBuilder);
        waiter.constructStarbucks();
        //get the drink built
        Starbucks drink = waiter.getstarbucksDrink();
    }
}
```

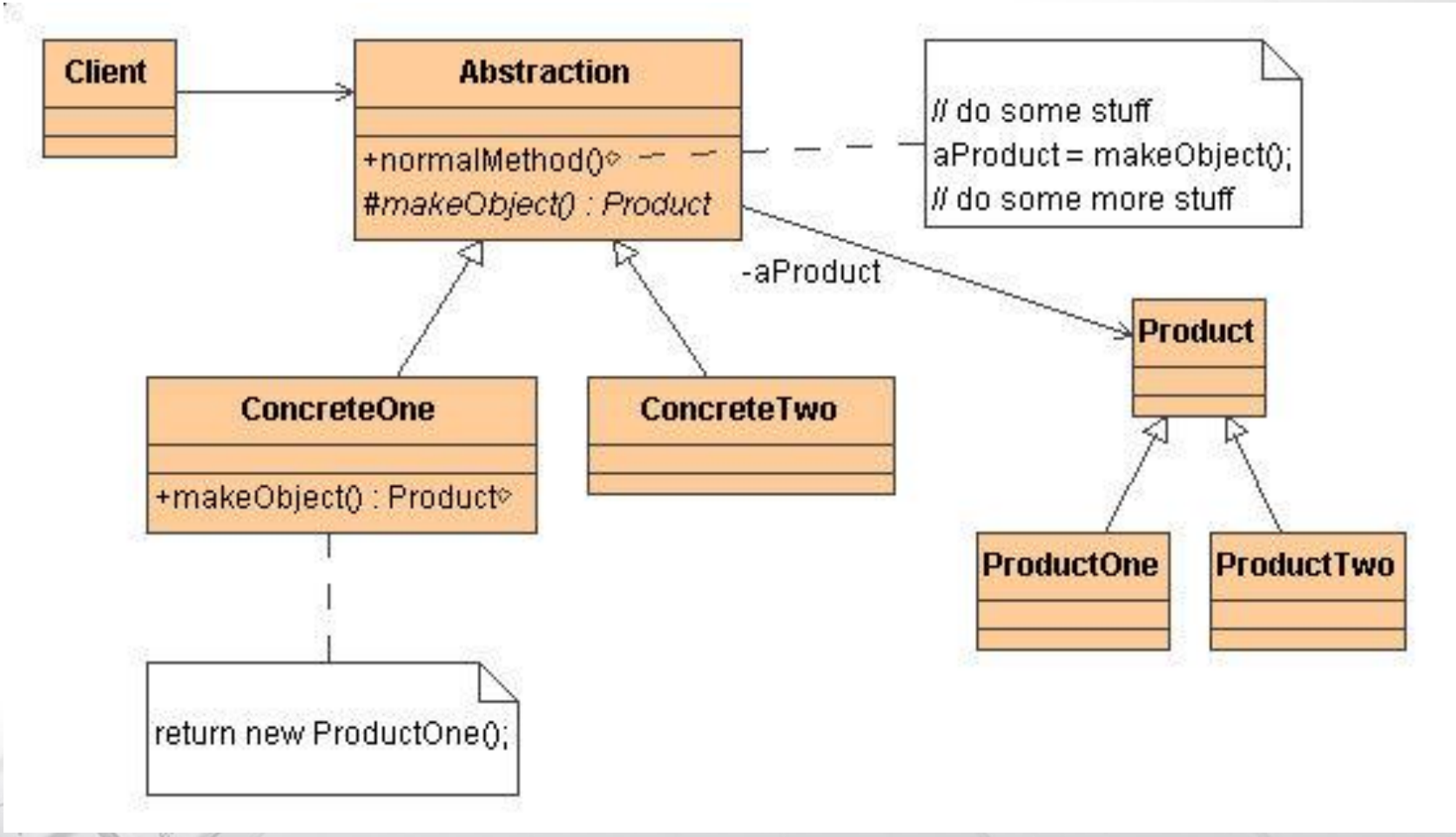
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Строитель / Builder

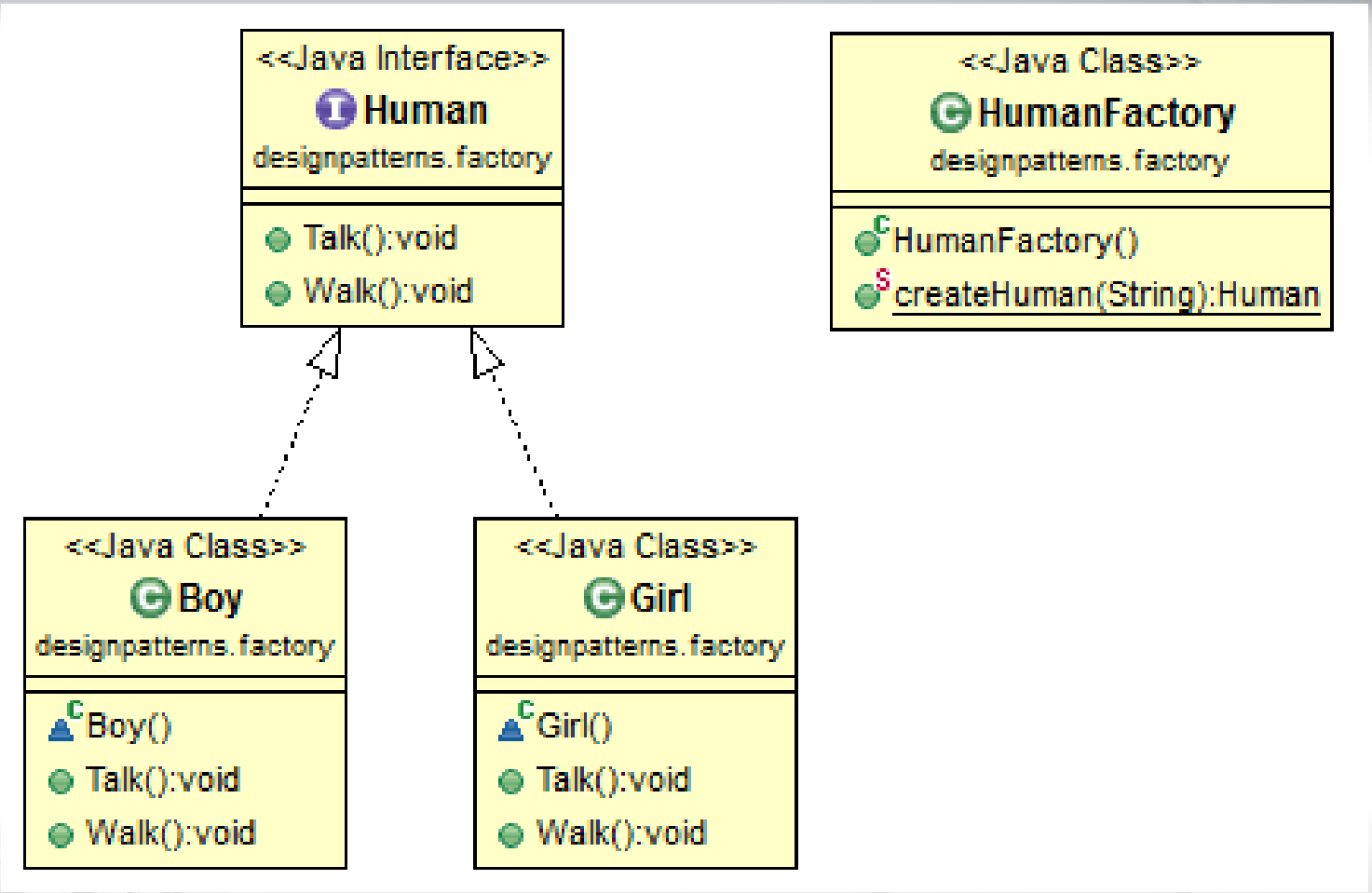
В JDK

```
StringBuilder strBuilder= new StringBuilder();
strBuilder.append("one");
strBuilder.append("two");
strBuilder.append("three");
String str = strBuilder.toString();
```

# Фабричный метод
# Factory method



# Фабричный метод
# Factory method

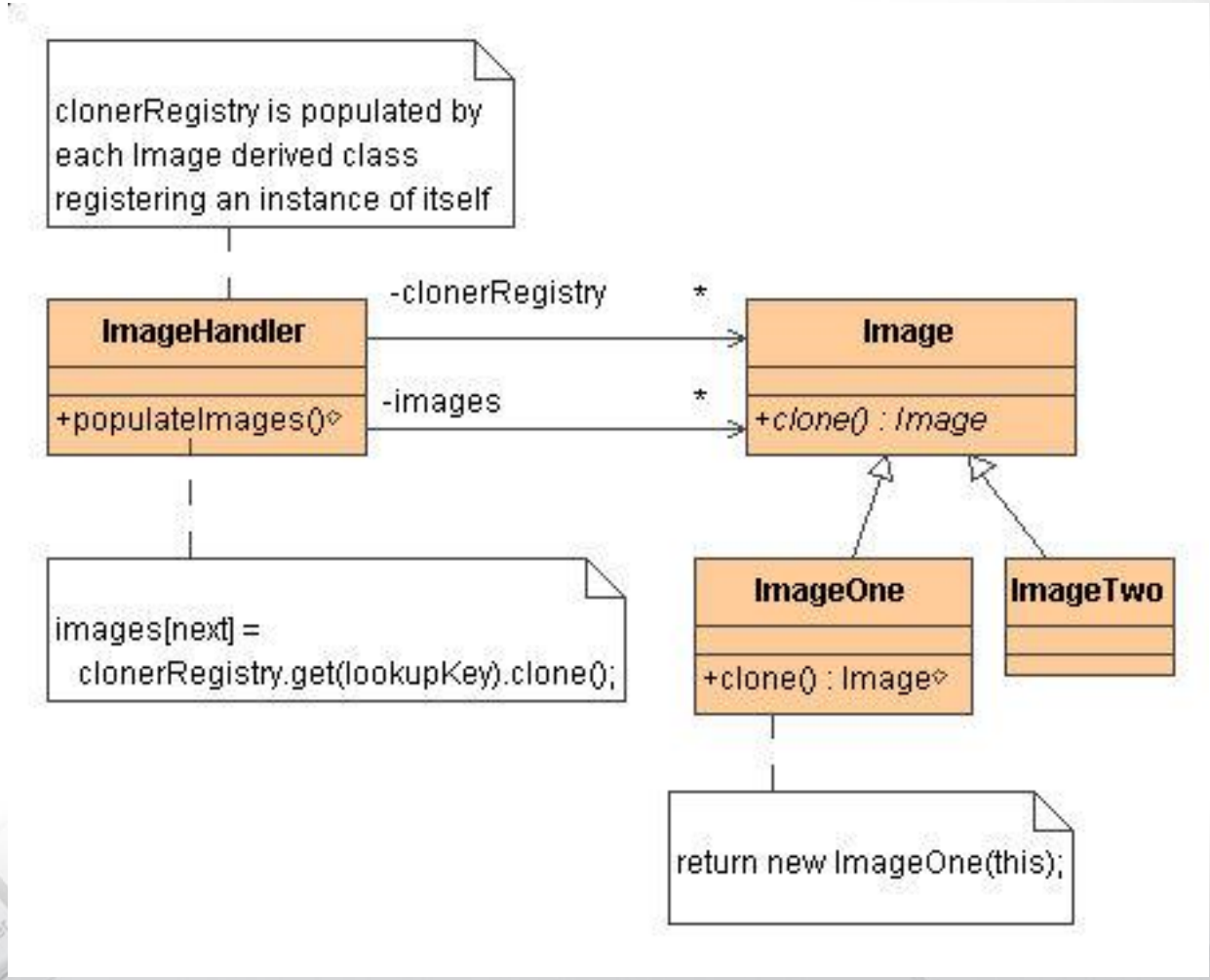


НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Фабричный метод Factory method

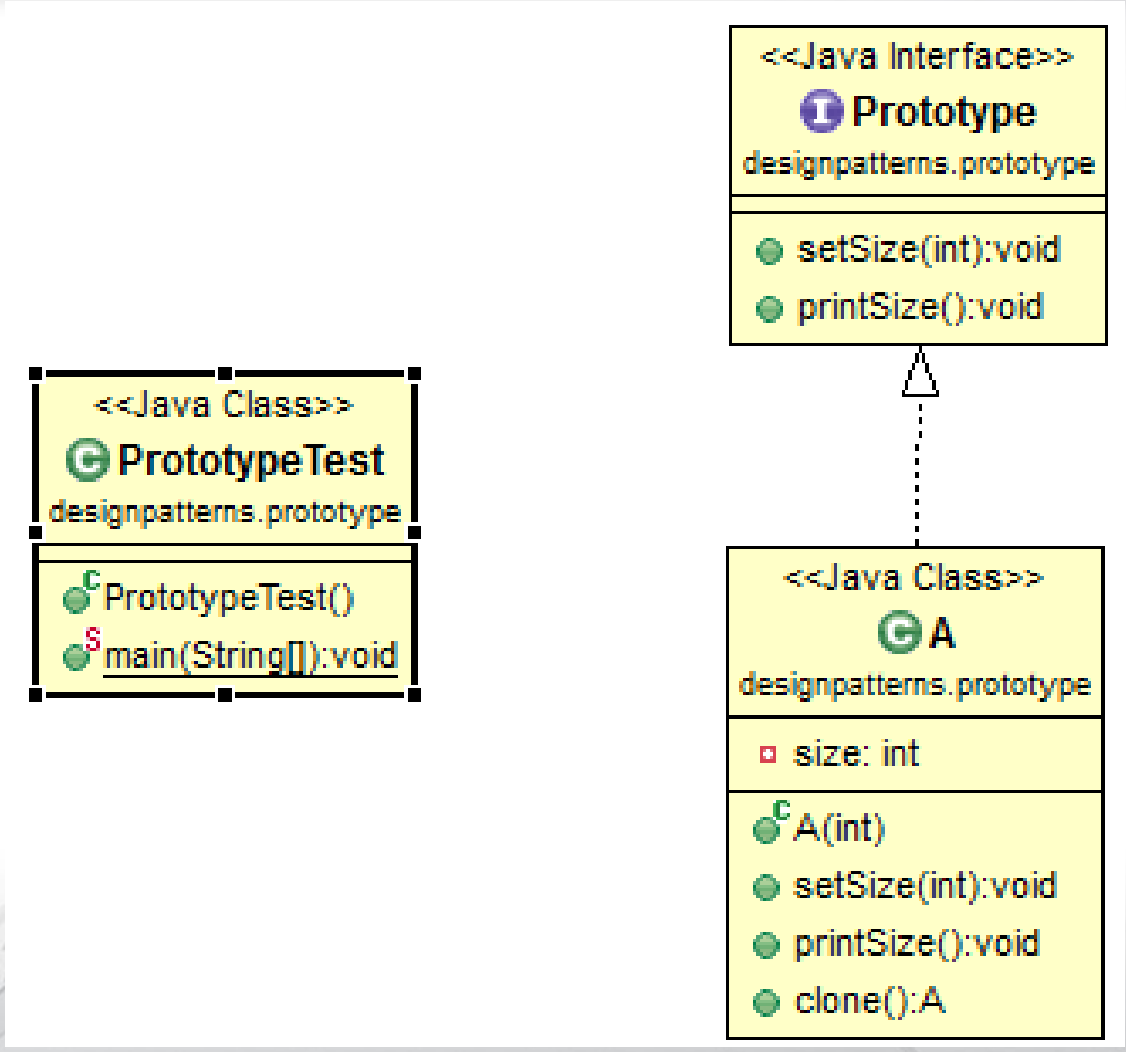
```
interface Human {
    public void Talk();
    public void Walk();
}
class Boy implements Human{
    @Override public void Talk() {
        System.out.println("Boy is talking...");
    }
    @Override public void Walk() {
        System.out.println("Boy is walking...");
    }
}
class Girl implements Human{
    @Override public void Talk() {
        System.out.println("Girl is talking...");
    }
    @Override public void Walk() {
        System.out.println("Girl is walking...");
    }
}
public class HumanFactory {
    public static Human createHuman(String m){
        Human p = m == "boy" ? new Boy() : new Girl();
        return p;
    }
}
```

# Прототип / Prototype



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Прототип / Prototype



НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Прототип / Prototype

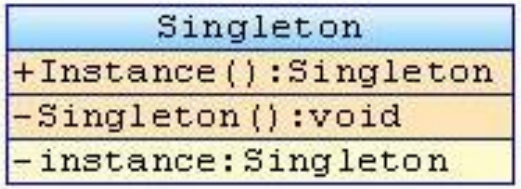
```
interface Prototype {
    void setSize(int x);
    void printSize();
}
class A implements Prototype, Cloneable {
    private int size;
    public A(int x) {
        this.size = x;
    }
    @Override public void setSize(int x) {
        this.size = x;
    }
    @Override public void printSize() {
        System.out.println("Size: " + size);
    }
    @Override public A clone() throws
                                CloneNotSupportedException {
        return (A) super.clone();
    }
}
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Прототип / Prototype

```
public class PrototypeTest {
    public static void main(String args[]) throws
                    CloneNotSupportedException {
        A a = new A(1);
        for (int i = 2; i < 10; i++) {
            Prototype temp = a.clone();
            temp.setSize(i);
            temp.printSize();
        }
    }
}
```

# Одиночка / Singleton

# Одиночка / Singleton

## Де два українці - там три гетьмани

- У страны может быть только один Президент (возможно, это нормально).
- Поэтому, когда нам нужен президент, просто используем AmericaPresident для получения его.
- Метод getPresident() гарантирует, что только один президент создан.
- В противном случае, это будет не очень хорошо.

# Одиночка / Singleton

Как этого добиться:

- private конструктор - ни один другой класс не может создать новый объект.
- private ссылка – внешняя модификация невозможна
- public static метод является единственным способом получения объекта.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Одиночка / Singleton

```
public class AmericaPresident {
    private AmericaPresident() {
    }
    private static AmericaPresident thePresident;
    public static AmericaPresident getPresident(){
        if(thePresident == null)
            thePresident = new AmericaPresident();
        return thePresident;
    }
}
```

# Одиночка / Singleton

## Пример использования

- java.lang.Runtime#getRuntime() - это часто используемый метод из стандартной бибиотеки Java.
- getRuntime() возвращает объект runtime ассоциированный с текущим приложением Java.
- Пример простого использования getRuntime(). Читаем web-страницу на Windows-системе.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Одиночка / Singleton

```
Process p = Runtime.getRuntime().exec(
    "C:/windows/system32/ping.exe www.nuos.edu.ua");
    //get process input stream and put it to
buffered reader
BufferedReader input = new BufferedReader(
      new InputStreamReader( p.getInputStream()));
String line;
while ((line = input.readLine()) != null) {
    System.out.println(line);
}
input.close();
```

# Объектный пул / Object pool

- **Объектный пул** (*object pool*) — набор инициализированных и готовых к использованию объектов.
- Когда системе требуется объект, он не создаётся, а берётся из пула.
- Когда объект больше не нужен, он не уничтожается, а возвращается в пул.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Объектный пул / Object pool

Если в пуле нет ни одного свободного объекта, возможна одна из трёх стратегий:

- Расширение пула.
- Отказ в создании объекта, аварийная остановка.
- В случае многозадачной системы, можно подождать, пока один из объектов не освободится.

# Ленивая инициализация Lazy initialization

- Ресурсоёмкая операция (создание объекта, вычисление значения) выполняется непосредственно перед тем, как будет использован её результат.
- Таким образом, инициализация выполняется «по требованию», а не заблаговременно.

# Ленивая инициализация Lazy initialization

- Частный случай ленивой инициализации — создание объекта в момент обращения к нему
- Как правило, он используется в сочетании с такими шаблонами как Factory method, Singleton и Proxy

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Порождающие паттерны

Вопросы?

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Порождающие паттерны

Спасибо!

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

# Паттерны (шаблоны) проектирования

## Порождающие паттерны

Евгений Беркунский
http://berkut.homelinux.com
eberkunsky@gmail.com