



Java

ОСНОВНИ ПОНЯТИЯ В Java

- основна структура на Java програма
- поток на изпълнение в Java
- основни типове данни
- управляващи конструкции

Лектор: Кирил Камбуров

Структура на Java програма

- Всичкият код трябва да бъде в клас
- Програмата представлява изпълним клас - трябва да съдържа main метод
- Чувствителност към главни и малки букви
- Всяко твърдение (statement) трябва да завършва с ;

Структура на Java програма

```
public class SimpleClass {  
    public static void main(String[] args) {  
        System.out.println("Hello!");  
    }  
}
```

Пример в NetBeans

Структура на Java програма

- Последователно изпълнение на твърденията
- Коментиране на кода
 - `// comment`
 - `/* comment */`
 - `/** javadoc */`

ПОТОК НА ИЗПЪЛНЕНИЕ В Java

```
public class ExecutionFlowExample1 {  
    public static void main(String[] args) {  
        System.out.println("Kiril"); /* will be  
                                     printed first */  
        System.out.println("Bozhidar");  
    }  
}
```

ПОТОК НА ИЗПЪЛНЕНИЕ В Java

```
public class ExecutionFlowExample2 {  
    public static void main(String[] args) {  
        System.out.println("Bozhidar"); // will be  
                                           // printed first  
        System.out.println("Kiril");  
    }  
}
```

Пример в NetBeans

Примитивни типове в Java

Целочислени типове

- `int` – 4 байта : -2,147,483,648 до 2,147,483, 647
- `short` – 2 байта : -32,768 до 32,767
- `long` – 8 байта – суфикс `L`
- `byte` – 1 байт : -128 до 127
- Могат да се използват 8, 10 и 16-тинни бройни системи

Примитивни типове в Java

Типове с плаваща запетая

- float – 4 байта – 6-7 значими знака след десетичната запетая – суфикс F
- double – 8 байта – 15 значими знака след десетичната запетая – суфикс D

Примитивни типове в Java

Символен тип

- `char` – 2 байта – съдържа един символ (пр. `'A'`, `'9'`, `'\n'`)
 - Поддържа UTF символи
 - `\n` – нов ред
 - `\r` – връщане в началото на реда
 - `\t` - таб

Примитивни типове в Java

Boolean тип

- `boolean` – има стойности `true` или `false`
 - не се преобразува в число
 - всички логически изрази дават като резултат `boolean`

Променливи в Java

- Всяка променлива трябва да има зададен тип и име
- Името трябва да започва с буква или `_` и да съдържа само букви, цифри и `_`
- Примери :
 - `double salary;`
 - `int vacationDays;`
 - `boolean isReady;`

Деклариране на променливи

- `int i;`
- `int i, q;`
- `int i, // counter 1`
`q, // counter 2`
`p; // result`
- `long daysLeftUntilEndOfTheYear;`
- `char mostCommonlyUsedCharacter;`

Инициализиране на променливи

- `int i; // declaration`
- `i = 5; // initialization`
- `int i = 5; // declaration + initialization`

Инициализиране на променливи

- Не инициализирани променливи не могат да се ползват!

```
int students;
```

```
System.out.println(students);
```

```
// ERROR - variable not initialized
```

Инициализиране на променливи

- Променливи могат да бъдат декларирани и инициализирани навсякъде

```
double salary = 65000.0;
```

```
System.out.println(salary);
```

```
int vacationDays = 12; // correct
```

Инициализиране на променливи

- На променливите могат да бъдат придавани нови стойности

```
int students = 10; // initial number of students
System.out.println(students); // prints 10
students = 20; // now we have 20 students
System.out.println(students); // prints 20
```

Пример в NetBeans

Оператори

- $+$ - събиране
- $-$ - изваждане
- $*$ - умножение
- $/$ - делене – целочислено, ако и двете числа са цели числа, иначе с плаваща запетая
- $\%$ - остатък при делене

Оператори

```
int i = 5 + 8; // initialize i to 13
```

```
i = i + 5; // add 5 to i, i becomes 18
```

```
i += 5; // add 5 to i, i becomes 23
```

```
int p = i + 8; // initialize to i plus 8, i.e. 31
```

```
int q = i + p; // initialize to i plus p, i.e. 54
```

```
int t = q * 7; // initialize to q * 7, i.e. 378
```

Оператори за увеличаване и намаляване с 1

- Могат да бъдат прилагани само върху променливи от целочислен тип.
- Променят стойността на променливата.
- `++var` – увеличава стойността на `var` с 1 преди да бъде ползвана
- `--var` – намалява стойността на `var` с 1, преди да бъде ползвана

Оператори за увеличаване и намаляване с 1

```
int i = 5;
```

```
++i;
```

```
System.out.println(i); // prints 6
```

```
System.out.println(++i); // prints 7
```

```
System.out.println(i); // prints 7
```

```
System.out.println(--i); // prints 6
```

```
System.out.println(i); // prints 6
```

Оператори за увеличаване и намаляване с 1

- `var++` - увеличава с 1 стойността на променливата, след като стойността е използвана
- `var--` - намалява с 1 стойността на променливата, след като стойността е използвана

Оператори за увеличаване и намаляване с 1

```
int i = 5;
```

```
i++;
```

```
System.out.println(i); // prints 6
```

```
System.out.println(i++); // prints 6
```

```
System.out.println(i); // prints 7
```

```
System.out.println(i--); // prints 7
```

```
System.out.println(i); // prints 6
```

Пример в NetBeans

Оператори за сравнение

- Ползват се върху сравними примитивни типове
 - `==` - проверка за равенство
 - `!=` - проверка за различие
 - `<` - по-малко
 - `<=` - по-малко или равно
 - `>` - по-голямо
 - `>=` - по-голямо или равно

Оператори за сравнение

5 != 6 // true

6 != 6 // false

6 == 6 // true

6 <= 6 // true

6 >= 7 // false

8 > 9 // false

9 < 7 // false

Оператори за сравнение

```
int i = 5;
```

```
int q = 5;
```

```
q == i // true
```

```
q != i // false
```

Логически оператори

- Логически израз се изчислява, докато не е сигурна неговата стойност (short circuit)
- `&&` - логическо И
- `||` - логическо ИЛИ
- `!` - логическо отрицание

Таблица на истинност

p	q	p && q (AND)	p q (OR)
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Логически изрази

```
int x = 2;
```

```
x != 0 && 10 / x > 5 // false
```

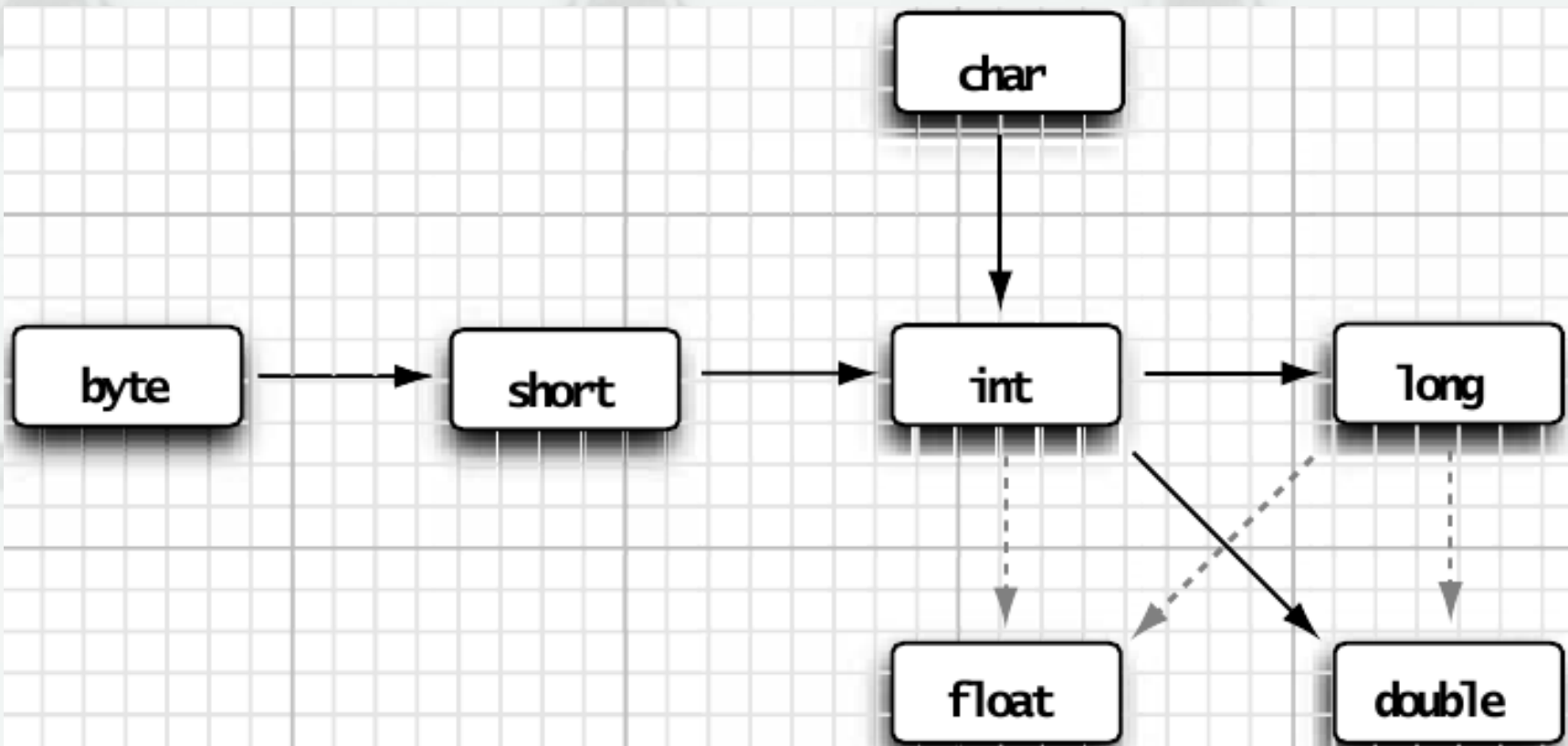
```
!(x > 5) && x != 3 // true
```

```
x > 5 || x == 8 // false
```

```
int y = 0;
```

```
(x < 5 || x > 9) && y != 0 && x + y != 0 // false
```

Автоматично преобразуване между примитивни типове



Изрично преобразуване (cast)

- Когато искаме преобразования водещи до загуба на информация
- (тип-към-който-преобразуваме) променлива

```
double x = 6.79;
```

```
int i = (int) x; // i == 6
```

```
long t = 8; int q = (int) t;
```

Пример в NetBeans

Блокове

- Позволява да бъдат групирани няколко твърдение
- Дефиницията на блок започва с { и завършва с } - между тях има твърдения
- Имат собствен обхват
- Могат да бъдат вложени един в друг

Блокове

```
public static void main(String[] args) {  
    int n;  
    ...  
    {  
        int k;  
        ... } // k is only defined up to here  
    }
```

Блокове

```
public static void main(String[] args) {  
    int n;  
    {  
        int k;  
        int n; // error--can't redefine n in inner  
              // block  
        ... }  
    }
```

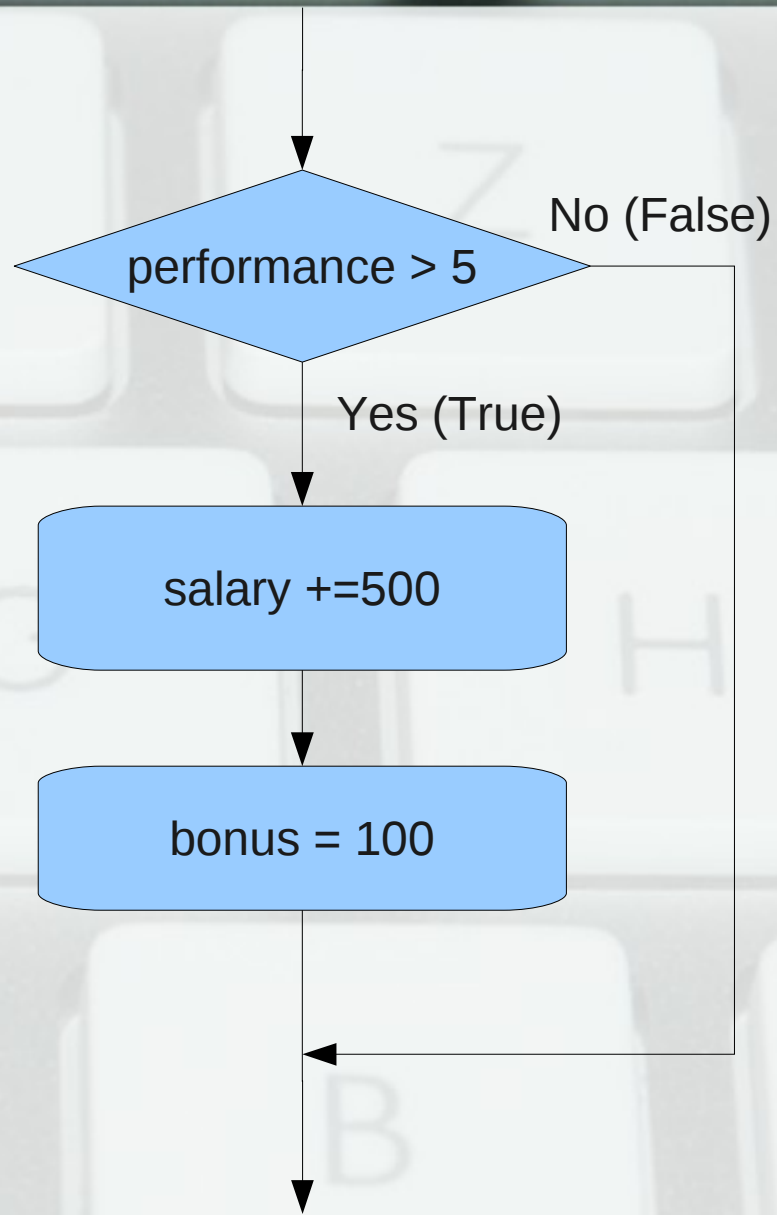
Пример в NetBeans

Условни конструкции – if

- if (condition) statement
 - condition – логически израз
 - statement – твърдение (или множество твърдения в блок), което да бъде изпълнено, ако условието е вярно

Условни конструкции - if

```
if (performance > 5){  
    salary+=500;  
    bonus = 100;  
}
```

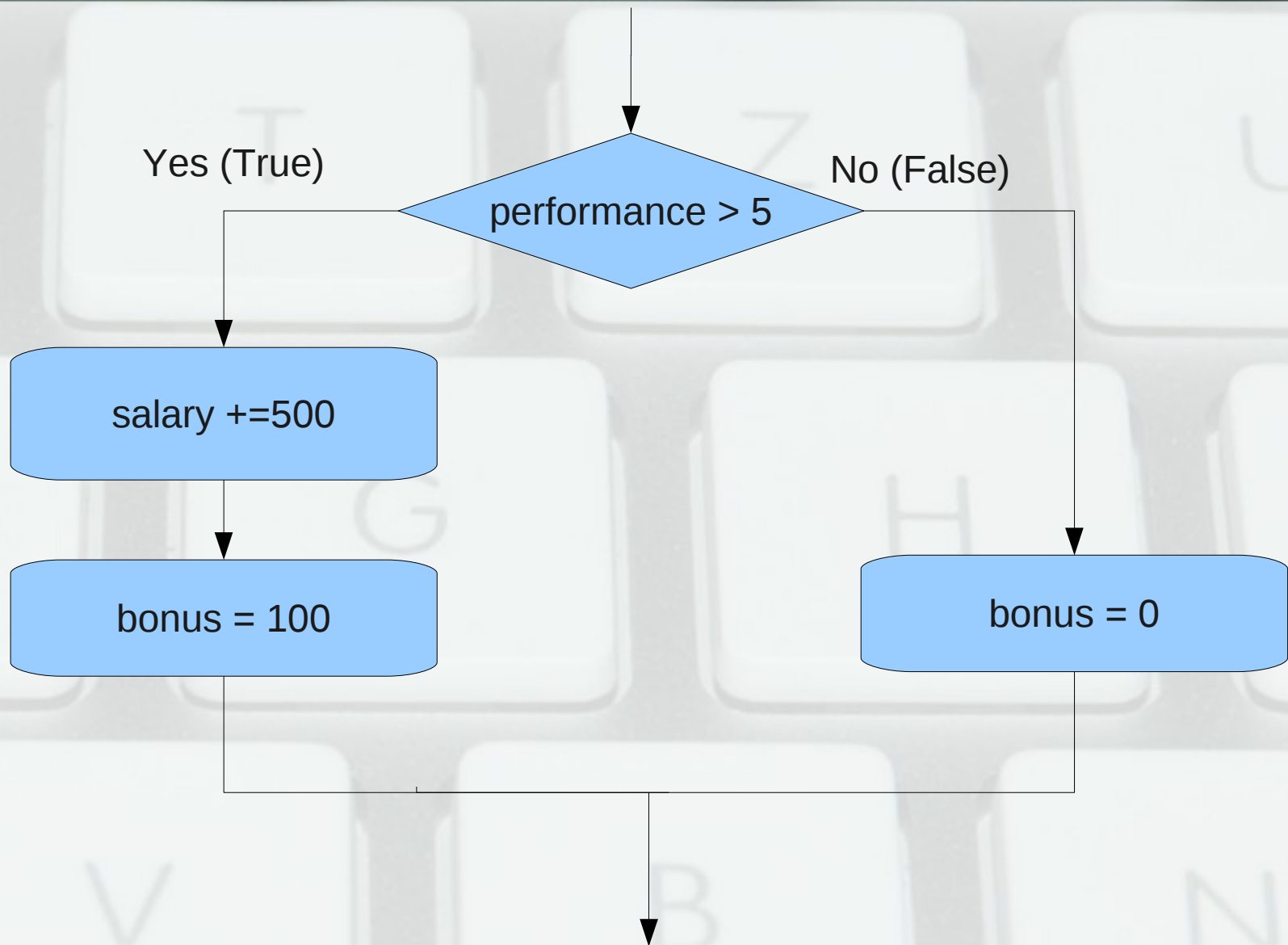


Условни конструкции – if-else

- if (condition) statement1 else statement2
 - condition – логически израз
 - statement1 – твърдение, което да бъде изпълнено, ако условието е вярно
 - statement2 – твърдение, което да бъде изпълнено, ако условието е невярно

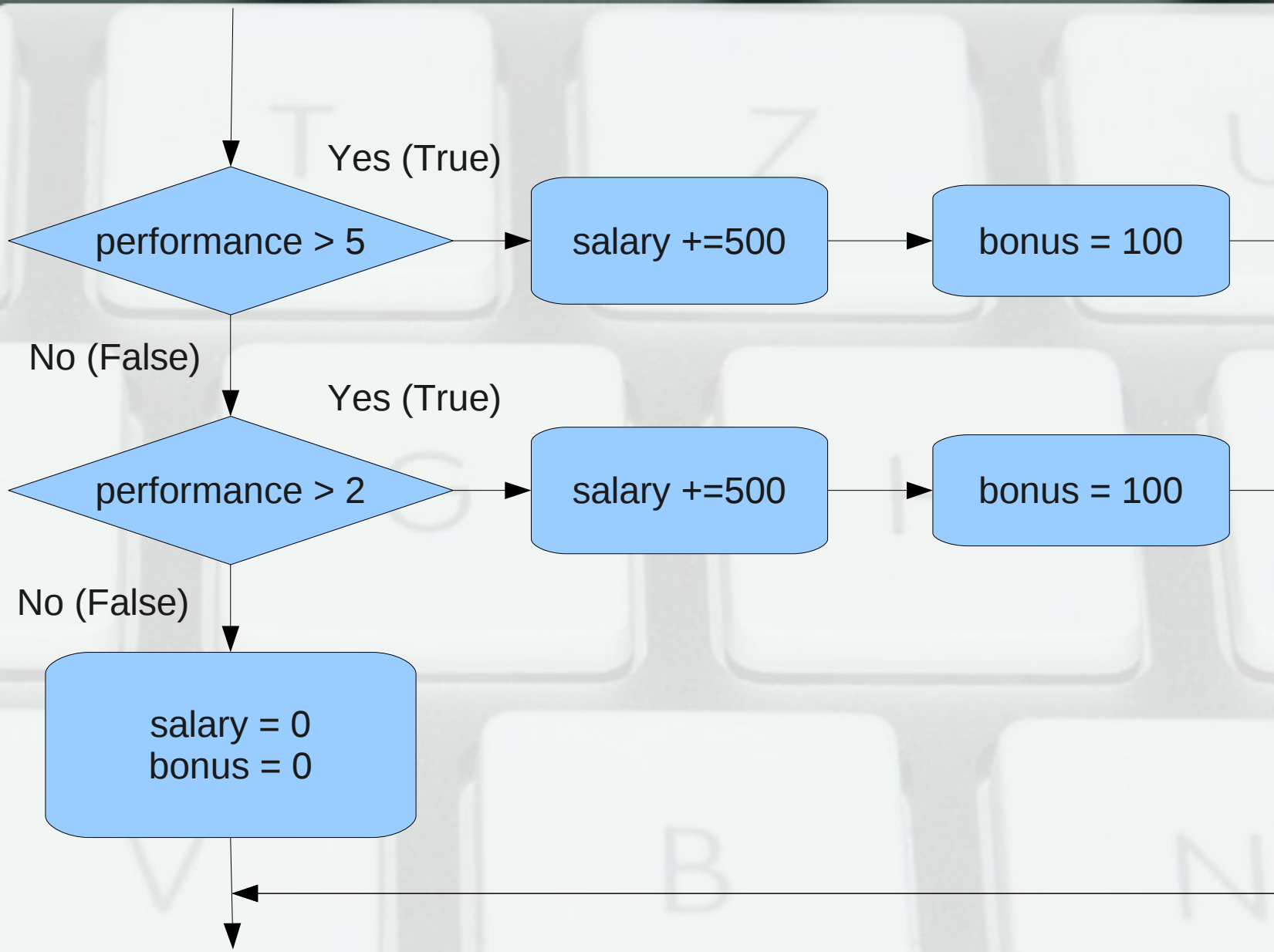
Условни конструкции – if-else

```
if (performance > 5){  
    salary += 500;  
    bonus = 100;  
} else {  
    bonus = 0;  
}
```



Условни конструкции if-else-if-else-...

```
If (performance > 5) {  
    salary += 500;  
    bonus = 100;  
} else if (performance > 2) {  
    salary += 100;  
    bonus = 0;  
} else { salary = 0; bonus = 0; } // you're fired
```



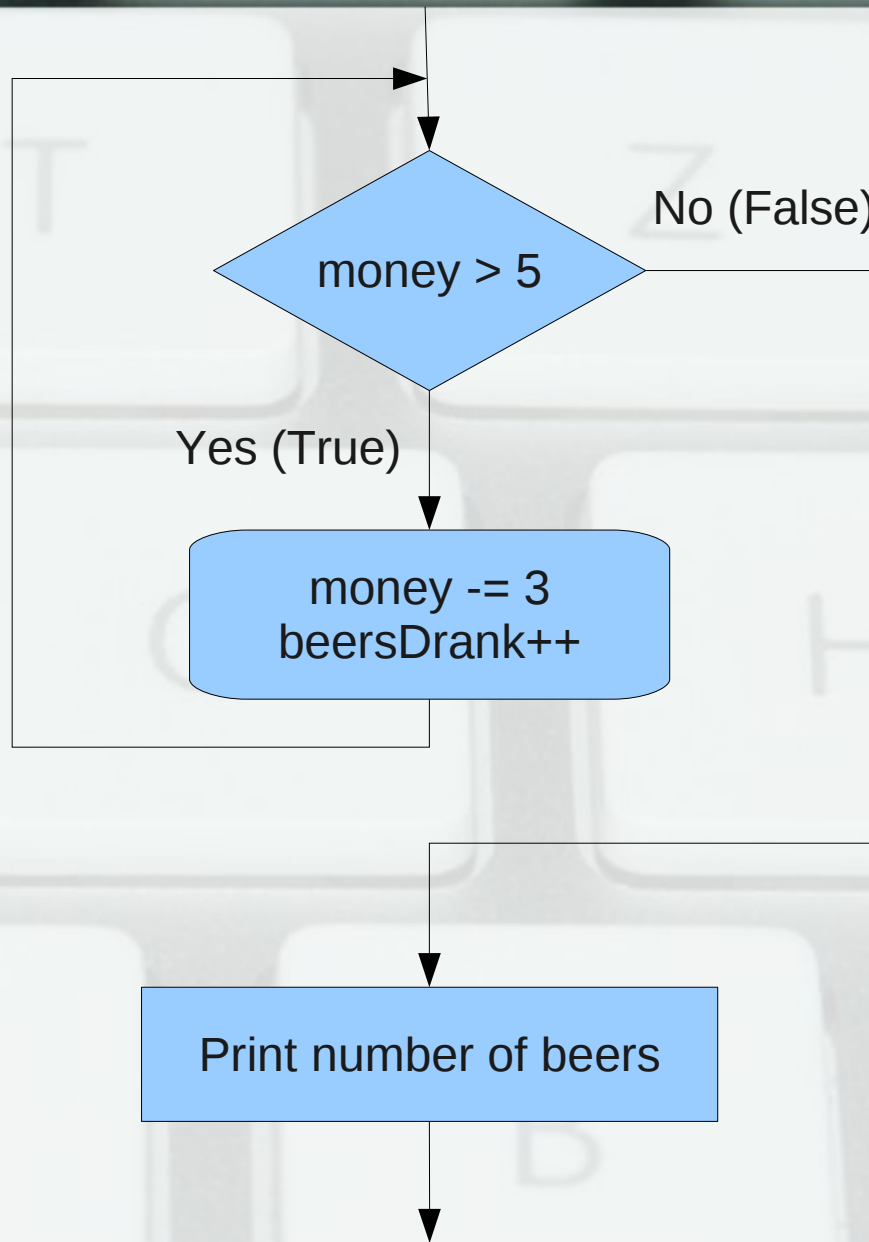
Пример в NetBeans

Цикли - while

- while (condition) statement
 - condition – логически израз
 - statement – твърдение, което се изпълнява, докато условието е вярно

Цикли - while

```
int beersDrank = 0;
while ( money > 5 ) {
    money -= 3; // buy beer
    beersDrank++;
}
System.out.print("We've drunk " +
    beersDrank + " beers.");
```



Цикли - do-while

- do statement while (condition)
 - statement – твърдение, което се изпълнява веднъж преди проверка на условието и след това, докато е вярно условието
 - condition – логически израз

Цикли - do-while

```
int count = 1;  
do {  
    System.out.println("Count is: " +  
        count);  
    count++;  
} while (count <= 11);
```

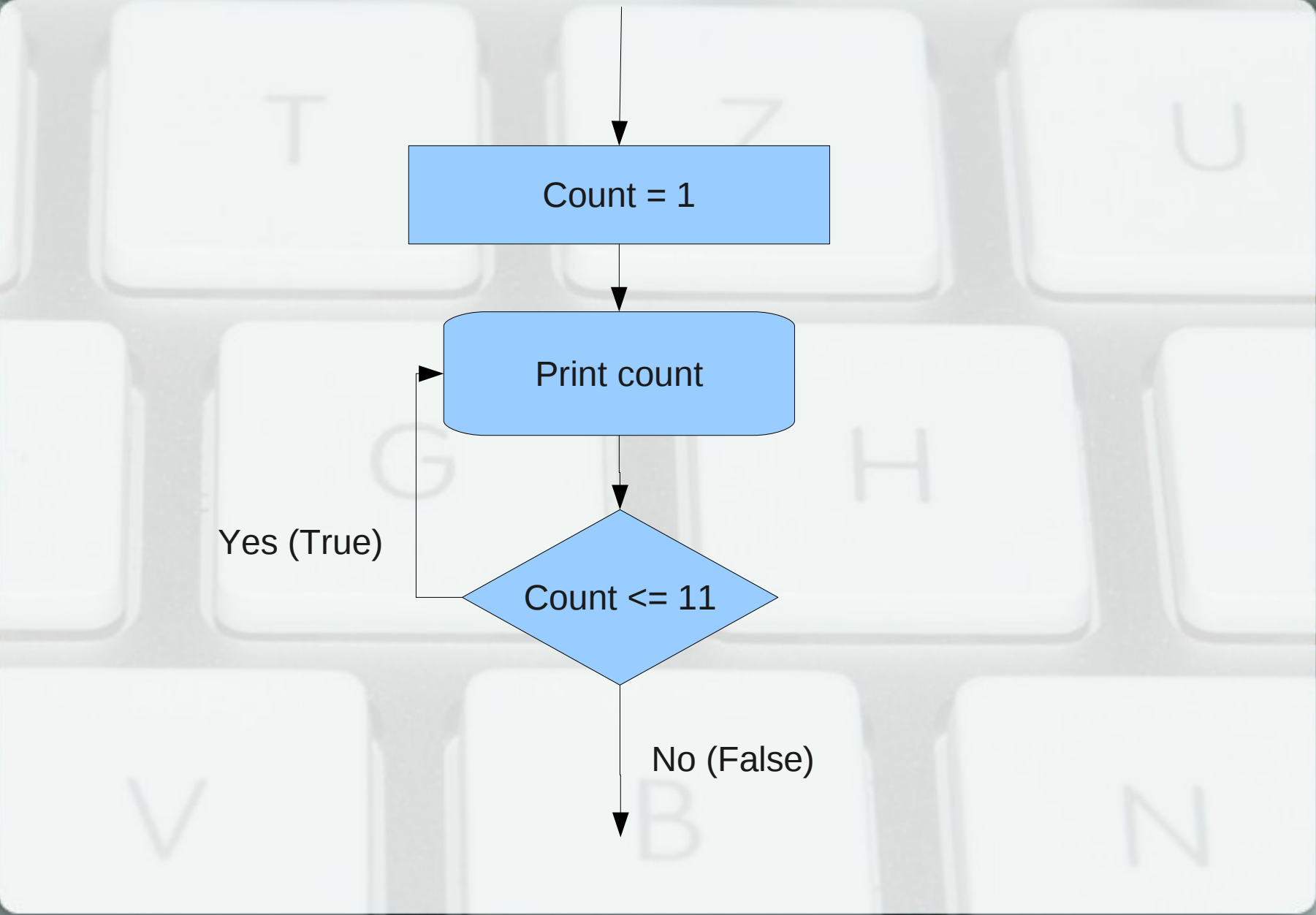
Count = 1

Print count

Count <= 11

Yes (True)

No (False)



Цикли - for

- for (initialization; condition; update)
statement
 - initialization – инициализираме променливи
 - condition – логически израз
 - update – изпълнява се, докато условието е вярно, преди statement
 - statement – изпълнява се, докато условието е вярно, след update

Цикли - for

```
for (int i = 1; i <= 10; i++) {  
    System.out.println(i);  
}  
// i no longer defined
```

Цикли - for

```
int i;  
for (i = 1; i <= 10; i++) {  
    System.out.println(i);  
}  
// i still defined
```

Пример в NetBeans

Множествен избор - switch

- Switch – множествен избор – може да се използва с byte, short, char, int

```
switch (variable) {  
    case VALUE : statements;  
    case VALUE : statements;  
    default : statements;  
}
```

```
int month = 8;
    switch (month) {
        case 1: System.out.println("January");
break;
        ...
        case 12:
System.out.println("December"); break;
        default: System.out.println("Invalid
month."); break;
    }
```

```
int startFrom = 6;
    switch (month) {
        case 1: System.out.println(1);
        ...
        case 6: System.out.println(6);
        case 10: System.out.println(10); break;
        default: System.out.println("Invalid
number."); break;
    }
```

Прекратяване на цикъл - break

- break – прекратява текущия цикъл. Може да се ползва в for, while, do-while и switch.

```
int i;
```

```
for( i = startNumber; i<startNumber+5;i++){
```

```
    if(i%5==0) break; }
```

```
System.out.println("closest number : " + i);
```

Преминаване към следваща итерация - continue

- continue – преминаване към следваща итерация в цикъл. Може да се използва в for, while, do-while цикли.

```
int numbersNotDivideableByThree = 0;
for(int i = startValue; i<1000; i++){
    if(i%3==0) continue;
    numbersNotDivideableByThree++; }
```

Пример в NetBeans

Масиви

- Масив – съдържа елементи от даден тип. Може да бъде многоизмерен (матрица). Достъп по индекс, индексацията започва от 0 (НУЛА). Чрез `.length` взимаме дължината на масива.
- `int[] integers` – масив от `int` елементи
- `char[] chars` – масив от `char` елементи

Масиви

```
String[] names = { "Pesho", "Gosho", "Ivan" };  
for(int i=0; i<names.length; i++){  
    System.out.println("Name : " + names[i]);  
}
```

Масиви

```
int[] numbers = { 1, 8, 10, 33 };  
long sum = 0;  
for(int i = 0; i < numbers.length; i++) {  
    sum += numbers[i];  
}  
System.out.println("Sum : " + sum);
```

For-each цикъл

- for (array-type variable : array) statement
 - Array-type – тип на променливата variable, трябва да съвпада с типа на елементите в масива
 - Variable – променлива, съдържаща текущия елемент от масива
 - Array – масив, от който да бъдат взимани елементи

For-each цикъл

```
String[] names = { "Pesho", "Gosho", "Ivan" };  
for(String name : names){  
    System.out.println("Name : " + name);  
}
```

Домашно

- Програма, която изважда следните СИМВОЛИ

*

* *

* * *

* * * *

* * * * *

Домашно

- Програма, която изважда следните СИМВОЛИ

* * * * *

* * * *

* * *

* *

*

Домашно

- Програма, която изважда следните СИМВОЛИ

*

* *

* * *

* * * *

* * * * *

Домашно – триъгълник на Паскал

Информация за "Триъгълник на Паскал"

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Домашно

- Пращайте решенията си на fosscourse@googlegroups.com
- За абониране към майл листа - <http://fosscourse.org/static/list-subscribe.html>