Java Basic

lecture #6. if-else-if

Mentor: <….>

lecture #6. if-else-if

- if
- if-else
- nested if
- if-else-if ladder
- Jump – break, continue, return

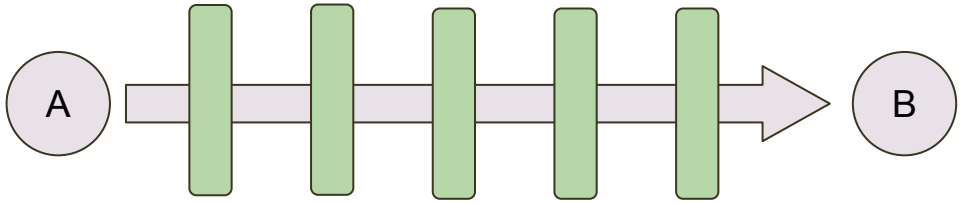
- Practice
  - Разные варианты условий (выбор валюты)
  - Выбор языка (ru, en, ua, de)

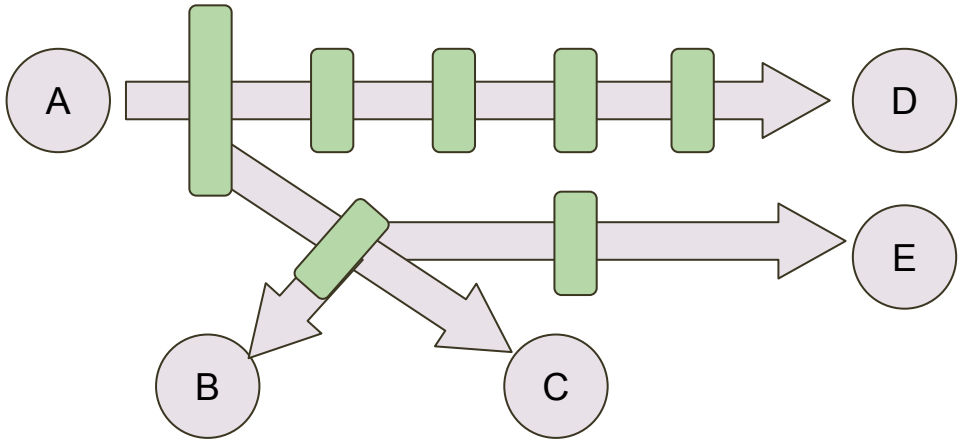
Принятие решений в Java

Принятие решений в программировании похоже на принятие решений в реальной жизни.
В программировании также встречаются ситуации, когда мы хотим, чтобы определенный блок кода выполнялся при выполнении некоторого условия.

Язык программирования использует операторы управления для управления потоком выполнения программы на основе определенных условий.
Они используются для того, чтобы поток выполнения продвигался и разветвлялся в зависимости от изменений состояния программы.

Операторы выбора Java:
if
if-else
nested-if
if-else-if
jump – break, continue, return





Оператор if

Оператор if является самым простым оператором принятия решения.
Он используется для принятия решения о том, будет ли выполняться определенный оператор или блок операторов, т. е. если определенное условие истинно, то блок операторов выполняется, в противном случае нет.

Синтаксис:

```
if(условие)
{
   // Операторы для выполнения
}
```

Например, если у нас есть свободное время на ланч - мы идем на ланч

Оператор if-else

Оператор if говорит нам, что если условие истинно, он выполнит блок операторов, а если условие ложно, то нет.
Но что, если мы хотим сделать что-то еще, если условие ложно.
Мы можем использовать оператор else с оператором if для выполнения блока кода, когда условие ложно.

Синтаксис:

```
if (условие)
{
    // Выполняет этот блок, если условие истинно
}
else
{
    // Выполняет этот блок, если условие ложно
}
```

Например, если у нас будет хорошая погода, то мы поедем кататься на лодке, иначе мы пойдем в ресторан

Оператор nested-if

Вложенный if – это оператор “если”, который является целью другого “если” или “еще”.
Вложенные операторы if означают оператор if внутри оператора if.
Java позволяет нам вкладывать операторы if в операторы if. т.е. мы можем поместить оператор if внутри другого оператора if.

Синтаксис:

```
if (условие1)
{
  // Выполняется, когда условие1 истинно
  if (условие2)
  {
    // Выполняется, когда условие2 истинно
  }
}
```

Например, если у нас будет хорошая погода то мы поедем за город, при этом если останется время - покатаемся на лошади

Оператор if-else-if ladder

Здесь выбираем один из нескольких вариантов. Операторы if выполняются сверху вниз. Как только одно из условий, управляющих if, становится истинным, выполняется оператор, связанный с этим if, и остальная часть лестницы игнорируется. Если ни одно из условий не выполняется, будет выполнен последний оператор else.

Синтаксис:

```
if (condition)
{
   statement;
}
else if (condition)
{
   statement;
}
else
{
   statement;
}
```

Например, если будет плохая погода - останемся дома, будем смотреть фильм.
Иначе если погода будет просто отличная - купим билеты и уедем в другой город на ярмарку.
Но если погода будет неопределенная, тогда встретимся с друзьями в нашем доме.

Jump – break, continue, return

Java поддерживает три оператора перехода: break, continue и return.
Эти три оператора передают управление другой части программы.

break - в основном используется для:

- Завершить последовательность в операторе switch.
- Чтобы выйти из цикла*

continue – используется для принудительного выполнения ранней итерации цикла. То есть вы можете захотеть продолжить выполнение цикла, но прекратить обработку остатка кода в его теле для этой конкретной итерации.

return - используется для явного возврата из метода. То есть управление программой передается обратно вызывающему методу.

*при этом break это не нормальное поведение выхода из цикла, нормальное поведение завершения цикла формируется в условном заголовке цикла, а оператор break используется лишь для прерывания цикла только тогда, когда возникают особенные ситуации.

Оператор “break”

Использование оператора break для выхода из цикла:

```
…
int[] a = {1,2,3,4,5,6,7,8,9,10};

for (int i = 0; i < a.length; i++) {

      if (a[i] == 5) {

            break;  // выход из цикла for

      } else {

            System.out.println(a[i]);

      }
}
```

Оператор “continue”

Использование оператора continue:

```
…
int[] a = {1,2,3,4,5,6,7,8,9,10};

for (int i = 0; i < a.length; i++) {

	System.out.print(a[i] + " ");

	if (a[i] % 2 == 0) {
		continue;
	}

	System.out.println("Не делится без остатка");
}
```

Тернарный оператор

оператор как альтернатива if-else конструкциям

Синтаксис:

variable = (condition) ? if true : if else;

для примера, алгоритм по определению агрегатного состояния по температуре (больше 0):

```
…
int temperature = scanner.nextInt();
String tempCondition = "";

if (temperature > 100) {
	tempCondition = "Пар";
} else {
	tempCondition = "Вода";
}
```

эту же конструкцию можно записать с помощью тернарного оператора:

```
tempCondition = temperature > 100 ? "Пар" : "Вода";
```