

PRIMER JAVA

LECTURER: Ms. Tran Le Nhu Quynh

Email (for submit student's homework) :

quynhtranlenhu@gmail.com

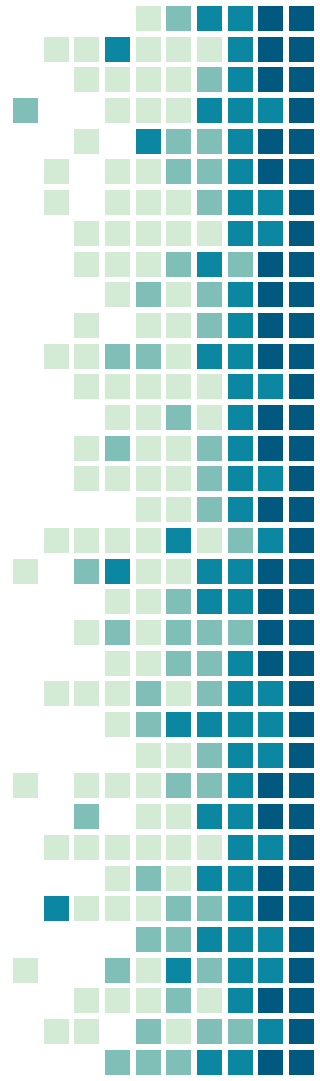
Email: nquynh@hcmuaf.edu.vn

Version : 2021- 2022



CONTENTS

1. Expressions
2. Control flow
3. Enum
4. Array
5. Junit Test
6. Scanner

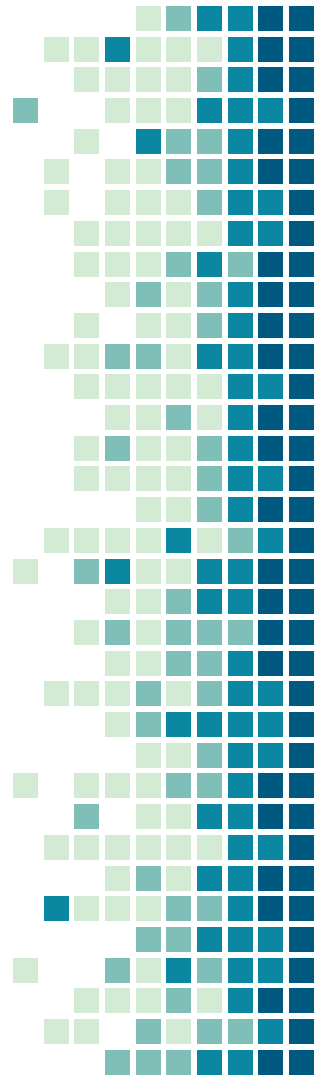




Expression

Literals

- A literal is any “constant” value that can be used in an assignment or other expression. Java allows the following kinds of literals:
- The **null** object reference (this is the only object literal, and it is allowed to be any reference type).
- Boolean: **true** and **false**.
- Integer: **int** type (example: 176 ; -15)
- Floating Point: **double** type (example: 3.245) , **float** type (example: 3.14E2)



Literals

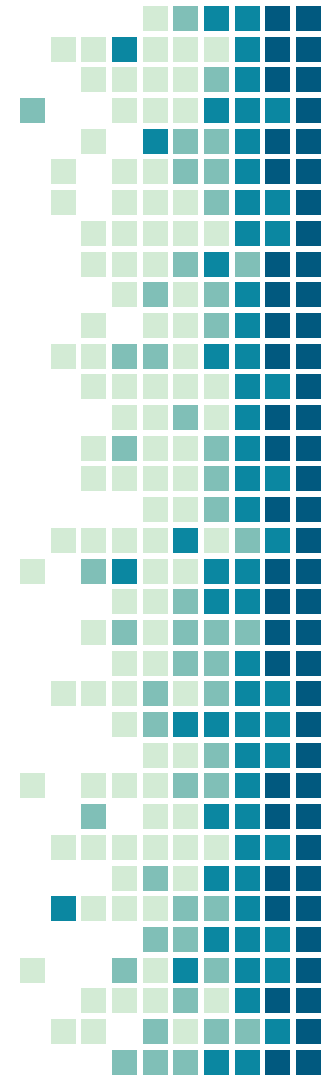
- Character: character constants are assumed to be taken from the Unicode alphabet. Typically, a character is defined as an individual symbol enclosed in single quotes. ‘ ‘
- Java defines the following special character constants:

<code>'\n'</code>	(newline)	<code>'\t'</code>	(tab)
<code>'\b'</code>	(backspace)	<code>'\r'</code>	(return)
<code>'\f'</code>	(form feed)	<code>'\\'</code>	(backslash)
<code>'\''</code>	(single quote)	<code>'\"'</code>	(double quote).

- String Literal: A string literal is a sequence of characters enclosed in double quotes “ “, for example, the following is a string literal: “Hello universe !”

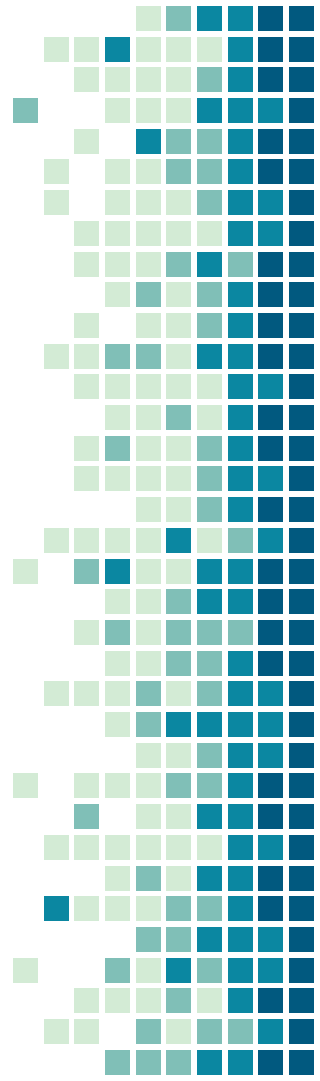
Operators

- Arithmetic operators
- String Concatenation
- Increment and Decrement Operators
- Logical Operators
- Bitwise Operators
- The Assignment Operator



Arithmetic operators

- + addition
- − subtraction
- * multiplication
- / division
- % the modulo operator



String Concatenation

- With strings, the (+) operator performs concatenation, so that the code

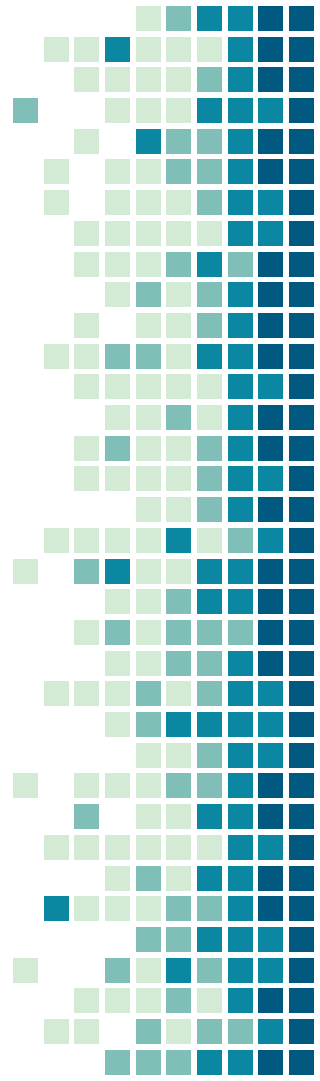
```
String rug = "carpet";
```

```
String dog = "spot";
```

```
String mess = rug + dog;
```

```
String answer = mess + " will cost me " + 5 + " hours!";
```

would have the effect of making answer refer to the string "carpetspot will cost me 5 hours!"



Increment and Decrement Operators

- the plus-one increment (++)
- the minus-one decrement (--)

```
int i = 8;  
int j = i++;           // j becomes 8 and then i becomes 9  
int k = ++i;           // i becomes 10 and then k becomes 10  
int m = i--;           // m becomes 10 and then i becomes 9
```

Logical Operators

< less than

<= less than or equal to

== equal to

!= not equal to

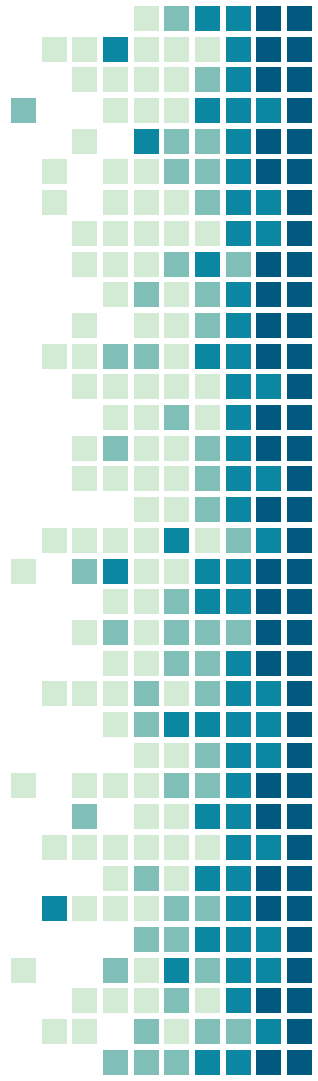
>= greater than or equal to

> greater than

! not (prefix)

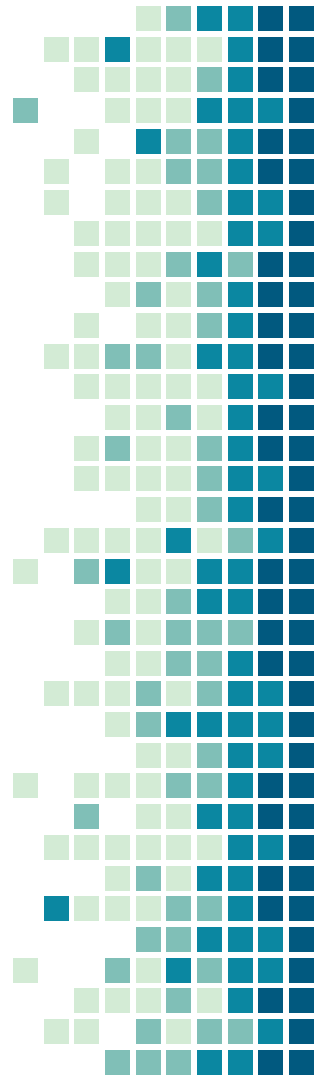
&& conditional and

|| conditional or



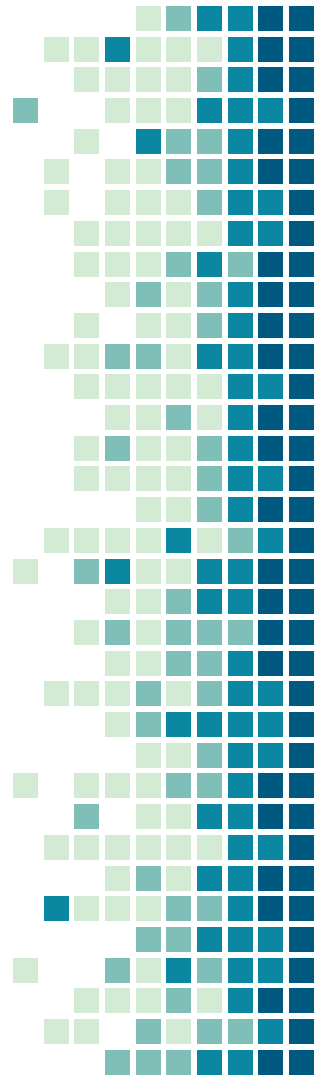
Bitwise Operators

- `~` bitwise complement (prefix unary operator)
- `&` bitwise and
- `|` bitwise or
- `^` bitwise exclusive-or
- `<<` shift bits left, filling in with zeros
- `>>` shift bits right, filling in with sign bit
- `>>>` shift bits right, filling in with zeros



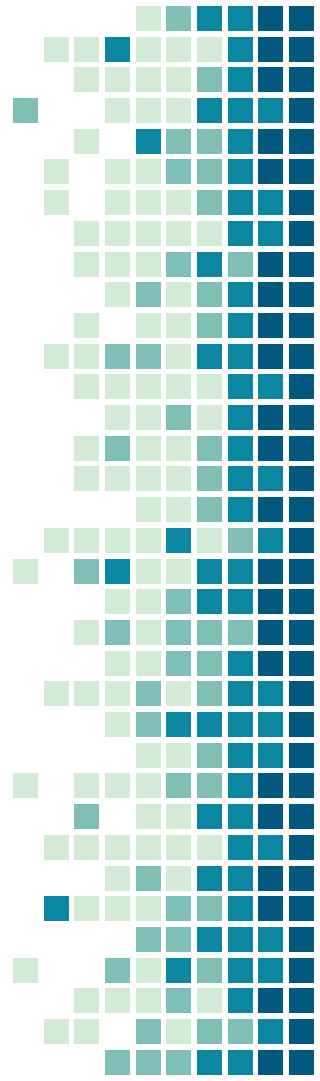
The Assignment Operator

- The standard assignment operator in Java is “=”
- `variable = expression`



Compound Assignment Operators

- variable op = expression
- variable = variable op expression
- Example:
- $x = x + 2 \Leftrightarrow x += 2$








Exercise 2.1

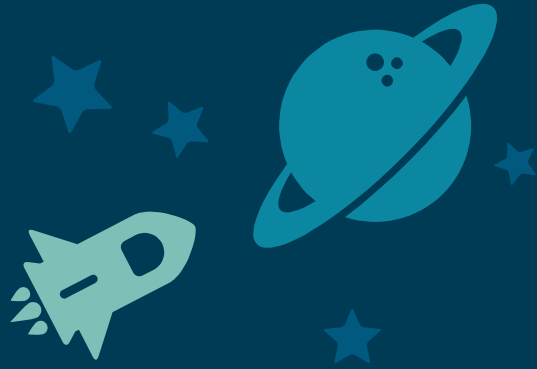
1. Create package: `utils`
2. Create class: `ShapeFormula`
3. Create methods to calculate area and perimeter for shapes. These methods can be called by Class Name.
4. Test main
5. JUnit test



Exercise 2.1

Shape	Formulas for Area (A) and Circumference (C)
Triangle 	$A = \frac{1}{2}bh = \frac{1}{2} \times \text{base} \times \text{height}$
Rectangle 	$A = lw = \text{length} \times \text{width}$
Trapezoid 	$A = \frac{1}{2}(b_1 + b_2)h = \frac{1}{2} \times \text{sum of bases} \times \text{height}$
Parallelogram 	$A = bh = \text{base} \times \text{height}$
Circle 	$A = \pi r^2 = \pi \times \text{square of radius}$ $C = 2\pi r = 2 \times \pi \times \text{radius}$



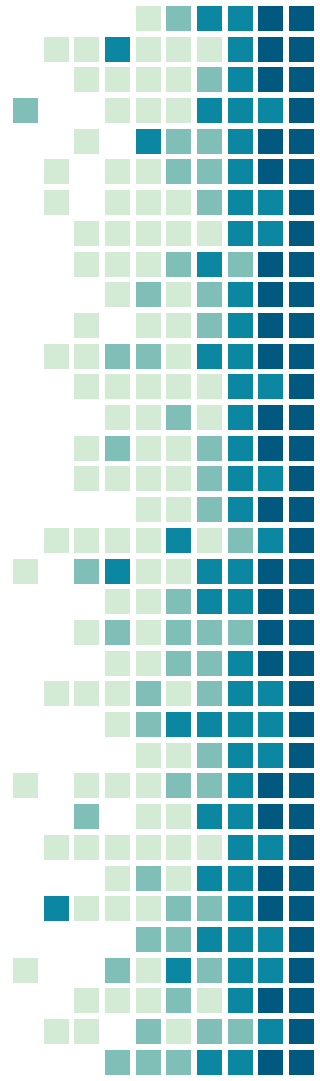


Control flow

“ . *Condition*

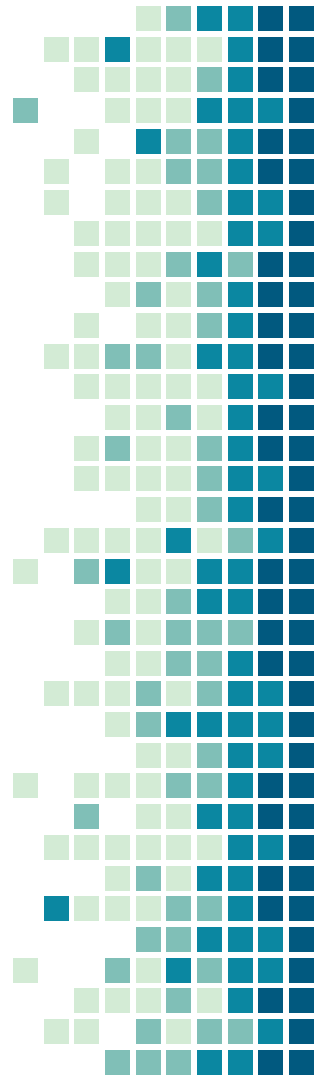
The If Statements syntax

```
if (booleanExpression)
    trueBody
else
    falseBody
```

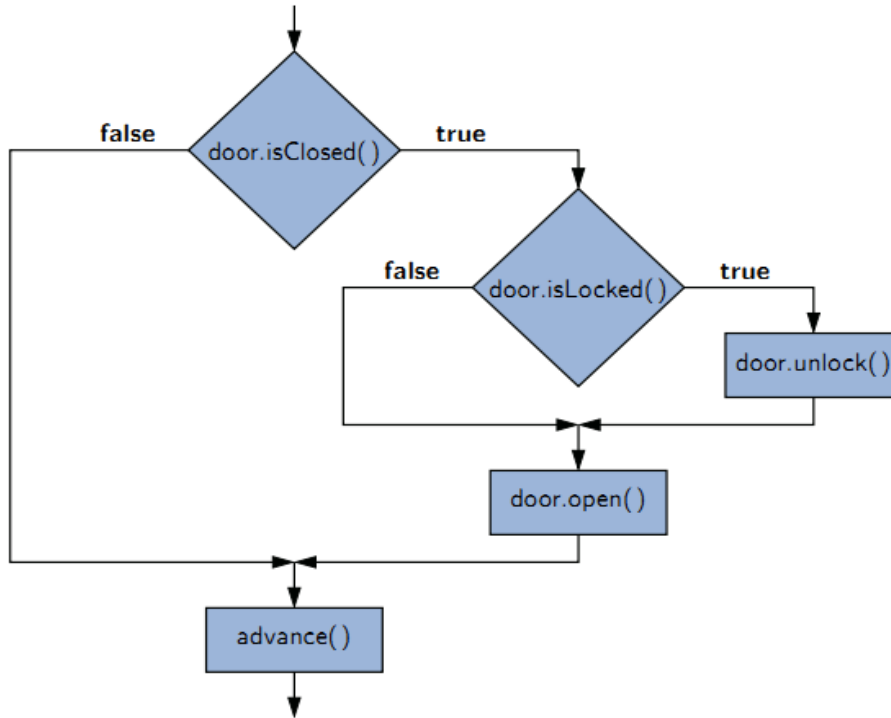


The If Statements syntax

```
if (firstBooleanExpression)
    firstBody
else if (secondBooleanExpression)
    secondBody
else
    thirdBody
```



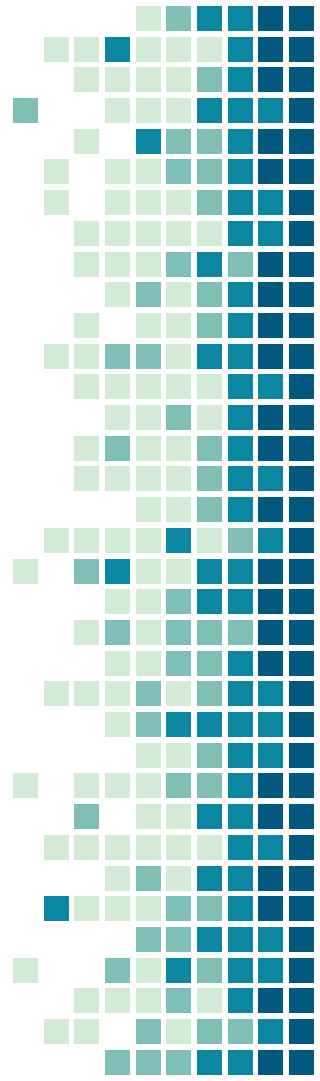
The If Statements example



```
if (door.isClosed( )) {  
  if (door.isLocked( ))  
    door.unlock( );  
  door.open( );  
}  
advance( );
```

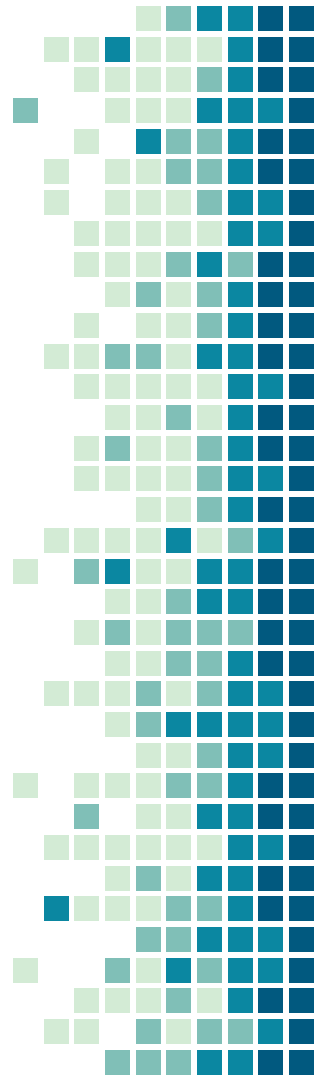
Java Switch Statements

- `switch(expression) {`
 `case x:`
 `// code block`
 `break;`
 `case y:`
 `// code block`
 `break;`
 `default:`
 `// code block`
 `}`



Java Switch Statements

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed.
- The break and default keywords are optional



Exercise 2.1

1. In package: utils
2. Create class: ArithmeticFormulas
3. isOddOrEven method with input: number (**int**) ; return **String**
 - Odd: 1,3,5,7,9...
 - Even: 2,4,6,8...
4. Exponential method with input: 2 number(**int**) ; return number(**int**) ;
 - Example: $2^2 = 4$
5. Factorial method:
6. Test main
7. JUnit test



Exercise 1.1

Question 4: create method: `classifyStudent()` in `Student` class

- Return String
- Follow requirements of next page
-



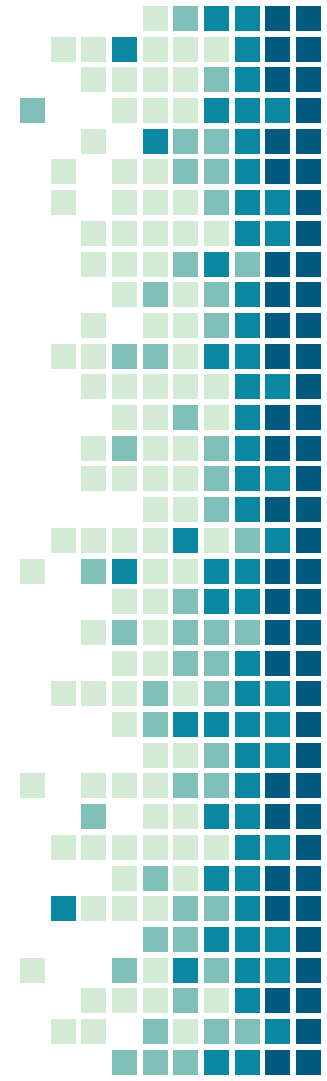
Exercise 1.1

- Loại giỏi:
 - + ĐTB các môn học từ 8,0 trở lên, trong đó ĐTB của 1 trong 2 môn Toán, Ngữ văn từ 8,0 trở lên; Không có môn học nào ĐTB dưới 6,5;
- Loại khá:
 - + ĐTB các môn học từ 6,5 trở lên, trong đó ĐTB của 1 trong 2 môn Toán, Ngữ văn từ 6,5 trở lên; Không có môn học nào ĐTB dưới 5,0;
- Loại trung bình:
 - + ĐTB các môn học từ 5,0 trở lên, trong đó ĐTB của 1 trong 2 môn Toán, Ngữ văn từ 5,0 trở lên; Không có môn học nào ĐTB dưới 3,5;
- Loại yếu:
 - + ĐTB các môn học từ 3,5 trở lên;
 - + Không có môn học nào ĐTB dưới 2,0.
- Loại kém: Các trường hợp còn lại.

“ . *Loop*

While Loops

- **while** (booleanExpression)
loopBody

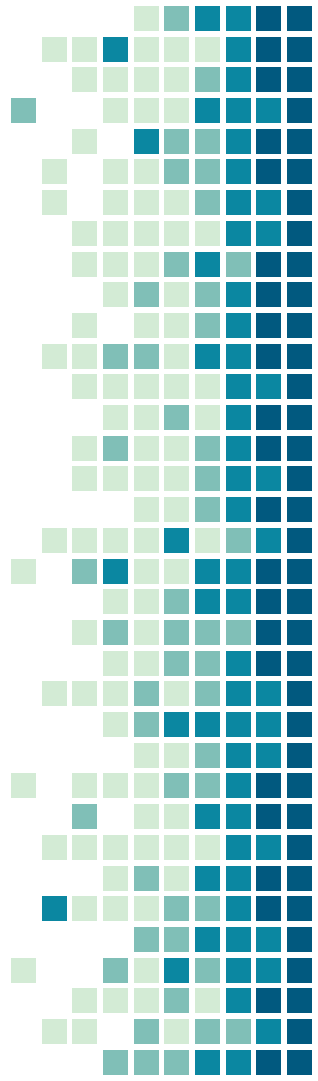


Do-While Loops

do

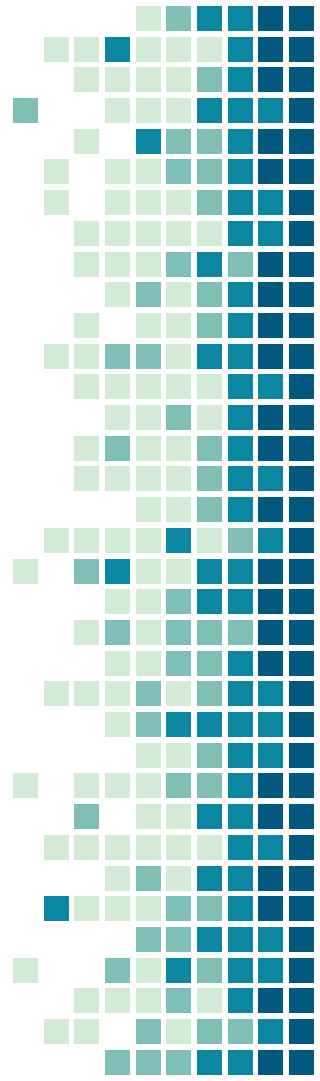
loopBody

while (booleanExpression)



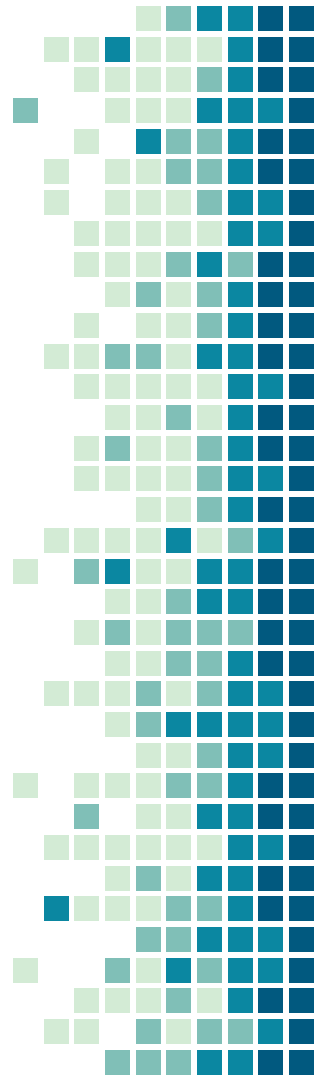
For loop

```
for (initialization; booleanCondition; increment)  
  loopBody
```



For-Each Loop

```
for (elementType name : container)  
loopBody
```





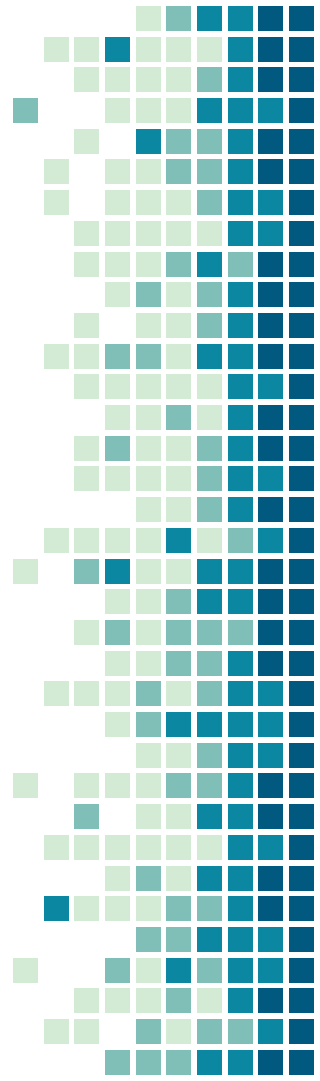
JUnit Test

Reading in JUnit Tutorial

Sr.No.	Methods & Description
1	void assertEquals(boolean expected, boolean actual) Checks that two primitives/objects are equal.
2	void assertTrue(boolean condition) Checks that a condition is true.
3	void assertFalse(boolean condition) Checks that a condition is false.
4	void assertNotNull(Object object) Checks that an object isn't null.
5	void assertNull(Object object) Checks that an object is null.
6	void assertSame(object1, object2) The assertEquals() method tests if two object references point to the same object.
7	void assertNotSame(object1, object2) The assertEquals() method tests if two object references do not point to the same object.
8	void assertEquals(expectedArray, resultArray); The assertEquals() method will test whether two arrays are equal to each other

Annotations

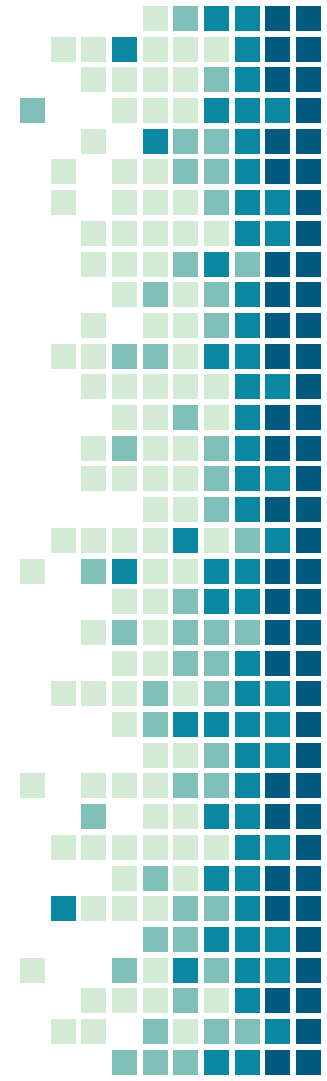
- Annotations are like meta-tags that you can add to your code, and apply them to methods or in class. These annotations in JUnit provide :
 - Methods are going to run before and after test methods.
 - Methods run before and after all the methods, and.
 - Methods or classes will be ignored during the execution.



Sr.No.	Annotation & Description
1	@Test The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.
2	@Before Several tests need similar objects created before they can run. Annotating a public void method with @Before causes that method to be run before each Test method.
3	@After If you allocate external resources in a Before method, you need to release them after the test runs. Annotating a public void method with @After causes that method to be run after the Test method.
4	@BeforeClass Annotating a public static void method with @BeforeClass causes it to be run once before any of the test methods in the class.
5	@AfterClass This will perform the method after all tests have finished. This can be used to perform clean-up activities.
6	@Ignore The Ignore annotation is used to ignore the test and that test will not be executed.

Exercise 1.1

- Create Junit test for `clasifyStudent ()` method
-

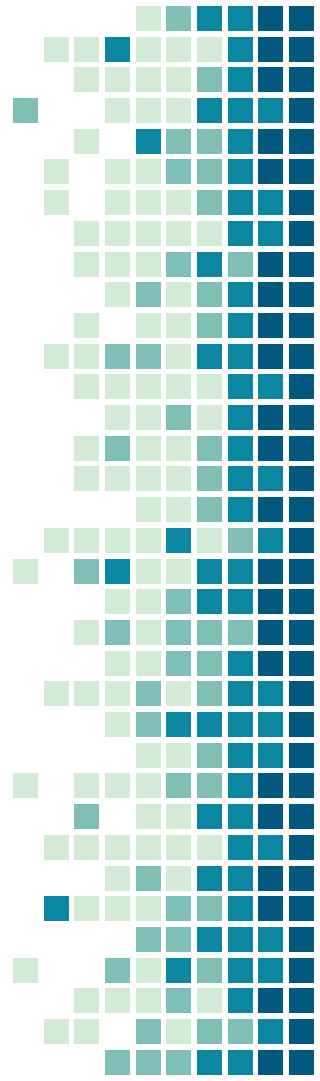




Scanner

Scanner

- Technically, the input is actually coming from the “standard in-put” device, which by default is the computer keyboard echoing its characters in the Java console. The `System.in` object is an object associated with the standard
- input device. A simple way of reading input with this object is to use it to create a
- Scanner object, using the expression **`new Scanner(System.in)`**



Scanner methods

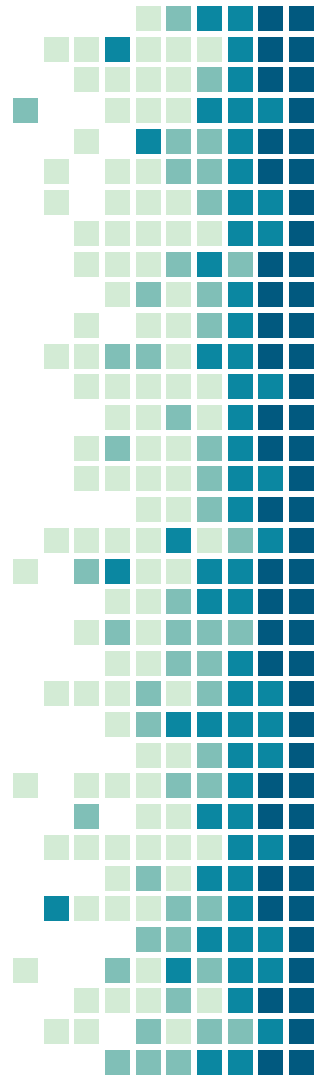
- **hasNext()**: Return true if there is another token in the input stream.
- **next()**: Return the next token string in the input stream; generate an error if there are no more tokens left.
- **hasNextType()**: Return true if there is another token in the input stream and it can be interpreted as the corresponding base type, Type, where Type can be Boolean, Byte, Double, Float, Int, Long, or Short.
- **nextType()**: Return the next token in the input stream, returned as the base type corresponding to Type; generate an error if there are no more tokens left or if the next token cannot be interpreted as a base type corresponding to Type.

Scanner methods

- **hasNextLine()**: Returns true if the input stream has another line of text.
- **nextLine()**: Advances the input past the current line ending and returns the input that was skipped.
- **findInLine(String s)**: Attempts to find a string matching the (regular expression) pattern s in the current line. If the pattern is found, it is returned and the scanner advances to the first character after this match. If the pattern is not found, the scanner returns null and doesn't advance.

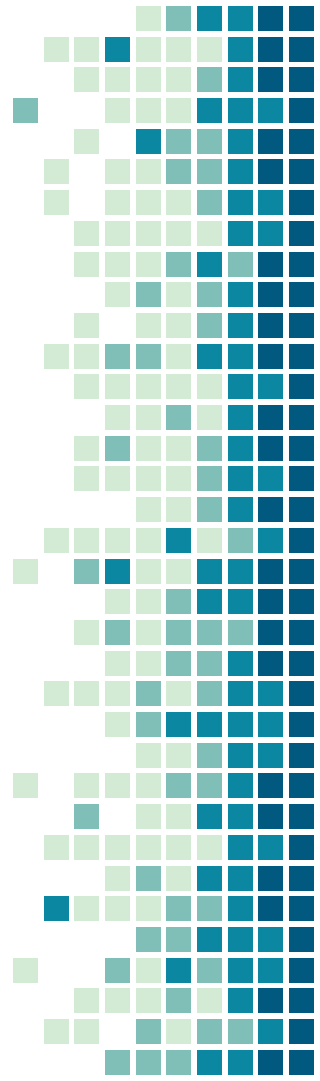
Scanner examples

```
public class InputExample {  
    public static void main(String[ ] args) {  
        Scanner input = new Scanner(System.in);  
        System.out.print("Enter your age in years: ");  
        double age = input.nextDouble( );  
        System.out.print("Enter your maximum heart rate: ");  
        double rate = input.nextDouble( );  
        double fb = (rate - age) * 0.65;  
        System.out.println("Your ideal fat-burning heart rate is " + fb);  
    }  
}
```



Scanner examples

```
public class InputExample {  
    public static void main(String[ ] args) {  
        Scanner input = new Scanner(System.in);  
        System.out.print("Enter your age in years: ");  
        double age = input.nextDouble( );  
        System.out.print("Enter your maximum heart rate: ");  
        double rate = input.nextDouble( );  
        double fb = (rate - age) * 0.65;  
        System.out.println("Your ideal fat-burning heart rate is " + fb);  
    }  
}
```



Exercise 2.2

1. In package: healthClinic
2. Create class: HealthDeclaration has attributes

```
private int id;  
private String fullName;  
private String address;  
private String birthDay;  
private double height;  
private double weight;  
private int systolic; // upper number of blood pressure  
private int diastolic; // lower number of blood pressure  
private final String hospitalName = "International Hospital";
```



Exercise 2.2

1. In package: healthClinic
2. Implements Comparable<HealthDeclaration>

- Write methods

```
/*
```

```
* BODY MASS INDEX (BMI) = WEIGHT / HEIGHT * HEIGHT
```

```
*/
```

```
public double getBodyMassIndex() {
```

```
return 0.0;
```

```
}
```



Exercise 2.2

- Write method
- `/*`
- `* Compare HealthDecleration by id`
- `* if id <another.id =>-99`
- `* if id > another.id =>99`
- `* equal = 0`
- `*/`
- `@Override`
- **`public int compareTo(HealthDecleration arg0) {`**
- **`// TODO Auto-generated method stub`**
- **`return 0;`**
- `}`



Exercise 2.2

- Write test by Scanner



Exercise 2.1

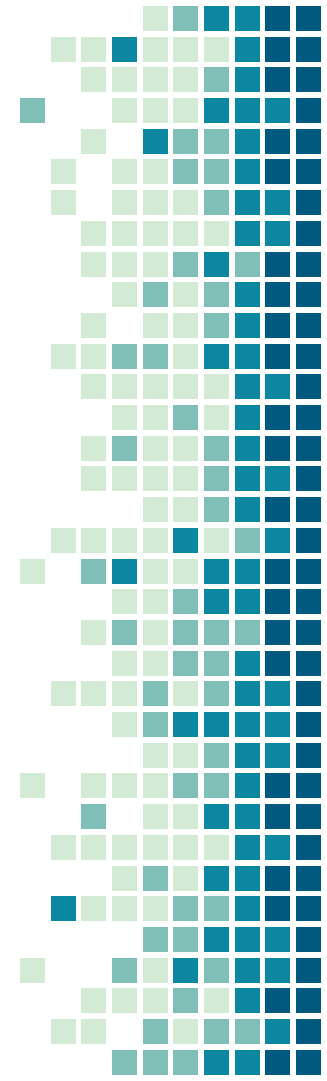
- In **utils** package
- Create **MoneyUtils** class
- Write test case for **classifyStudent()**





NOTES FOR NEXT WEEK

- **Review**
- **Do homework (work with computer and write down paper)**
- **Test 2 , 15 mins on class**
- String class in java collections
- Math class in java collections
- Date class in java collections
- Random
- Array



THANKS!