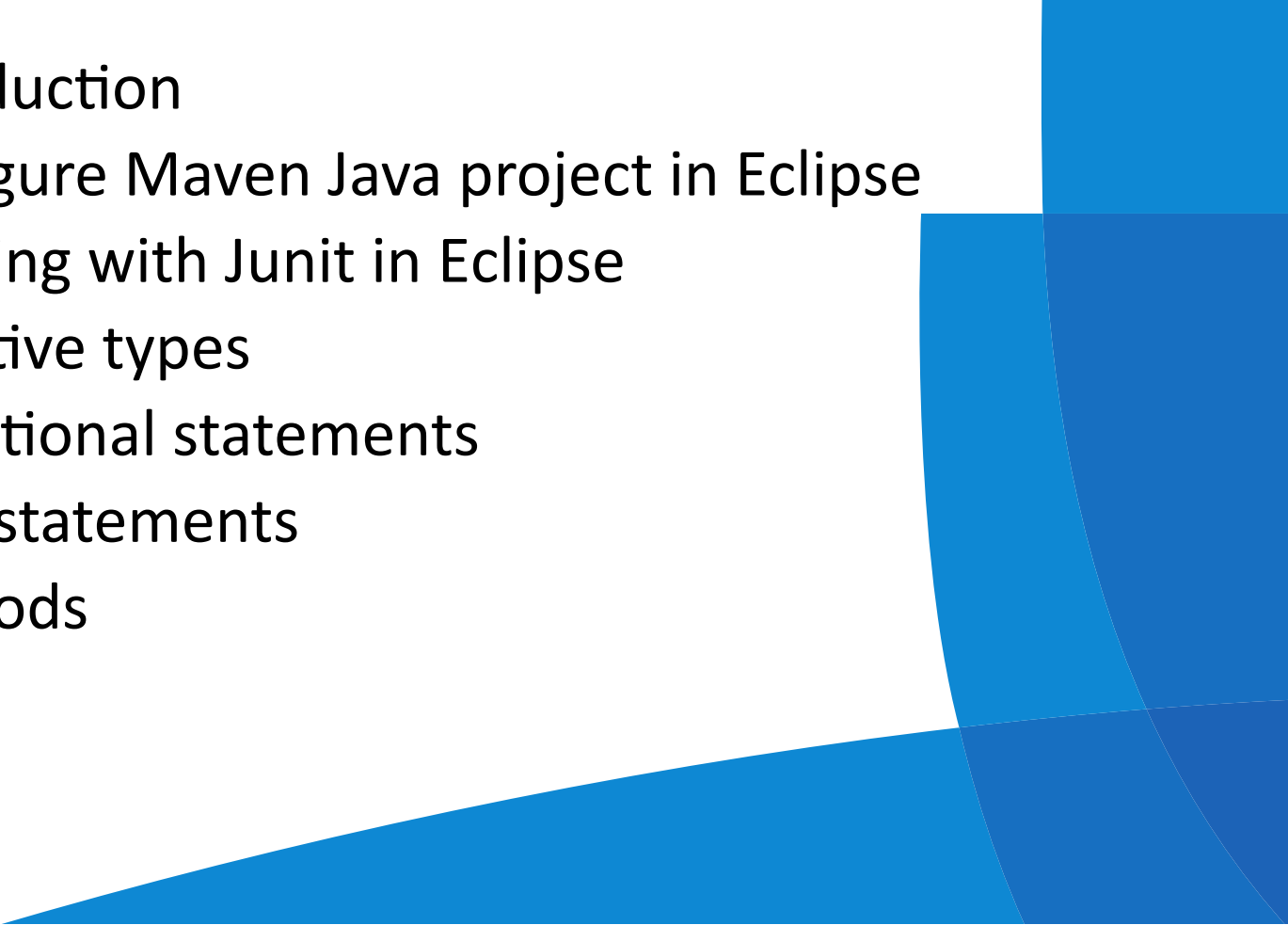


Agenda

- Introduction
 - Configure Maven Java project in Eclipse
 - Working with Junit in Eclipse
 - Primitive types
 - Conditional statements
 - Loop statements
 - Methods
- 

Who Am I?

- What's your name?
- What's your background?
- What's your function?
- Do you have any experience in programming?
- How used are you to classes?
- How are you doing on the smily thenometer?
- What do you hope to achieve in this course?

What is a computer

- Any ideas?
- Write your ideas on paper

What is programming?

- Write some ideas down on paper

Main parts of computer

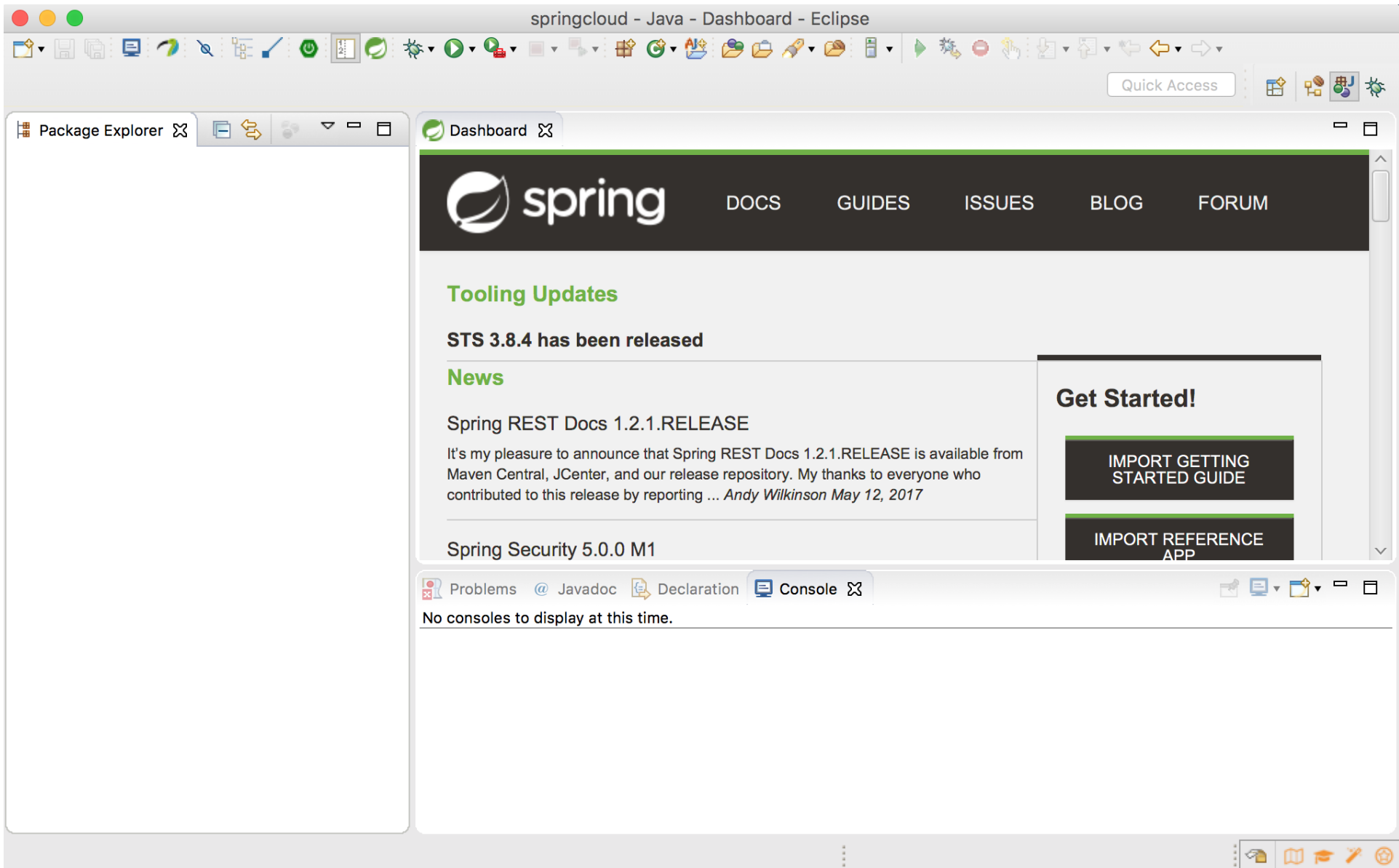
- processor
- memory
- interface

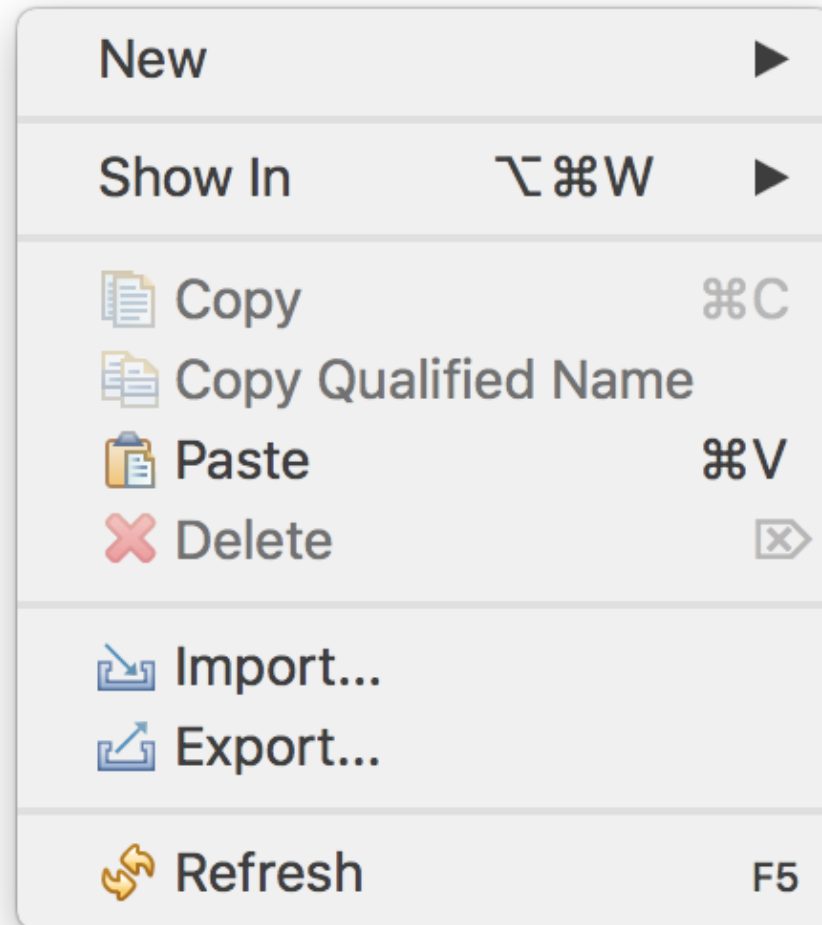
Working with Eclipse IDE

- Eclipse is a tools platform
 - -> large box to create tools
- One offspring is the Eclipse IDE
 - very versatile
 - modular
 - support different project formats

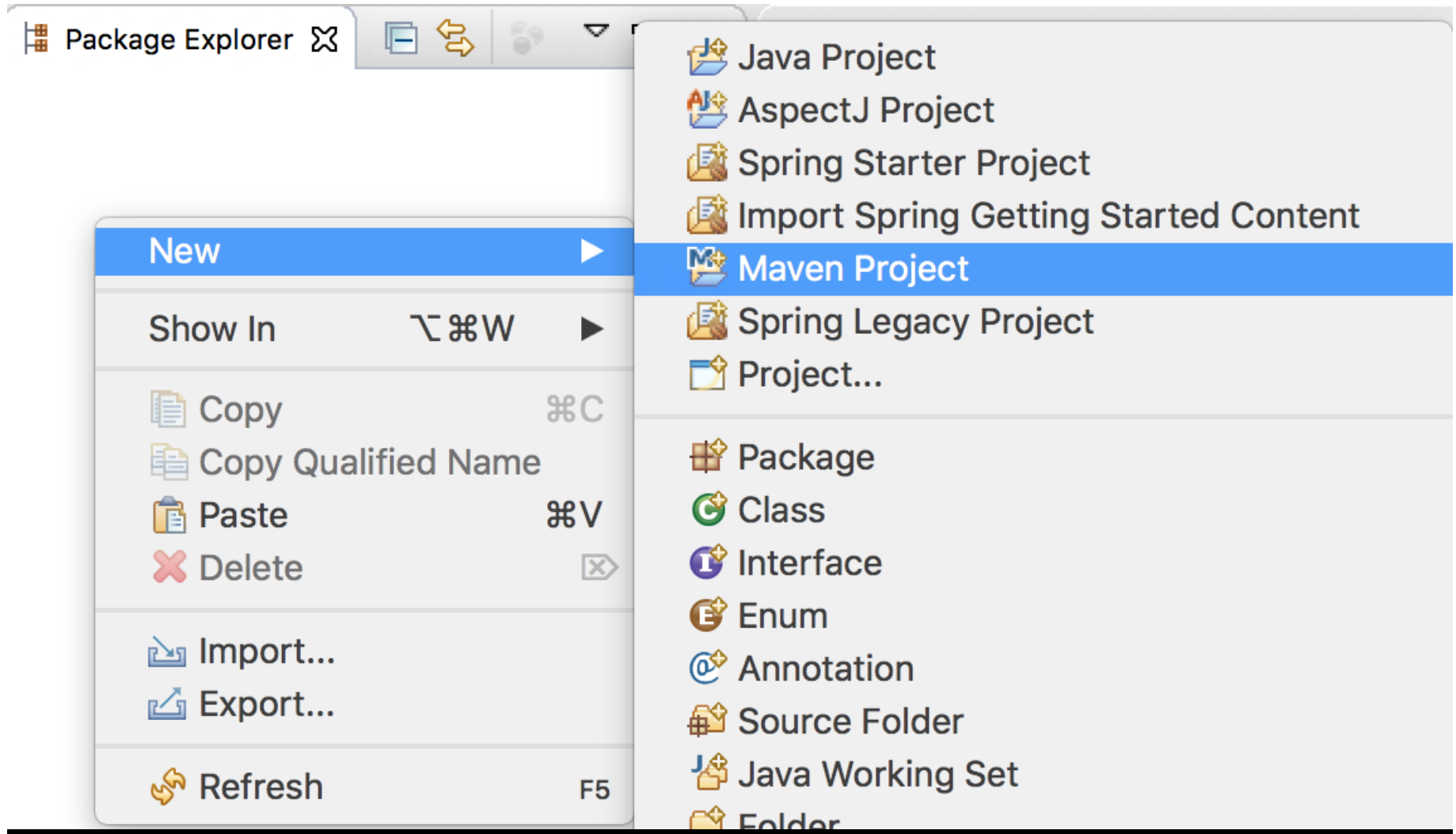
Maven Java Eclipse project

- Popular java project structure -> maven
- Maven is supported in eclipse
 - Some recurring configuration





Create in eclipse a maven project





New Maven Project

New Maven project

Select project name and location



☒ Create a simple project (skip archetype selection)

☒ Use default Workspace location

Location:

Browse...

☐ Add project(s) to working set

Working set:

More...

▶ **Advanced**



< Back

Next >

Cancel


Finish

New Maven Project

New Maven project

✖

Artifact id cannot contain spaces.



Artifact

Group Id:

nl.belastingdienst.courses.jft

▼

Artifact Id:

<your firstname>.dag0101.basics

▼

Version:

0.0.1-SNAPSHOT

▼

Packaging:

jar

▼

Name:

▼

Description:

Substitute for <your firstname> your firstname

Parent Project

Group Id:

▼

Artifact Id:

▼

Version:

▼

Browse...

Clear

▶ Advanced



< Back

Next >


Cancel

Finish

New Maven Project

New Maven project

Configure project



Artifact

Group Id:

nl.belastingdienst.courses.jft

▼

Artifact Id:

joris.dag0101.basics

▼

Version:

0.0.1-SNAPSHOT

▼

Packaging:

jar

▼

Name:

▼

Parent Project

Group Id:

▼

Artifact Id:

▼


Version:

▼

Browse...

Clear

▶ Advanced



< Back

Next >

Cancel

Finish


Package Explorer




joris.dag0101.basics

 src/main/java

 src/main/resources


 src/test/java

 src/test/resources

  JRE System Library [J2SE-1.5]

  src

 target

 pom.xml

Package Explorer



joris.dag0101.basics

 src/main/java

 src/main/resources

 src/test/java

 src/test/resources

  JRE System Library [J2SE-1.5]

  src

 target

 pom.xml

Change java 5 into java7 (or 8)

- By default a maven project is configured to use java 5
- Add properties to change this:

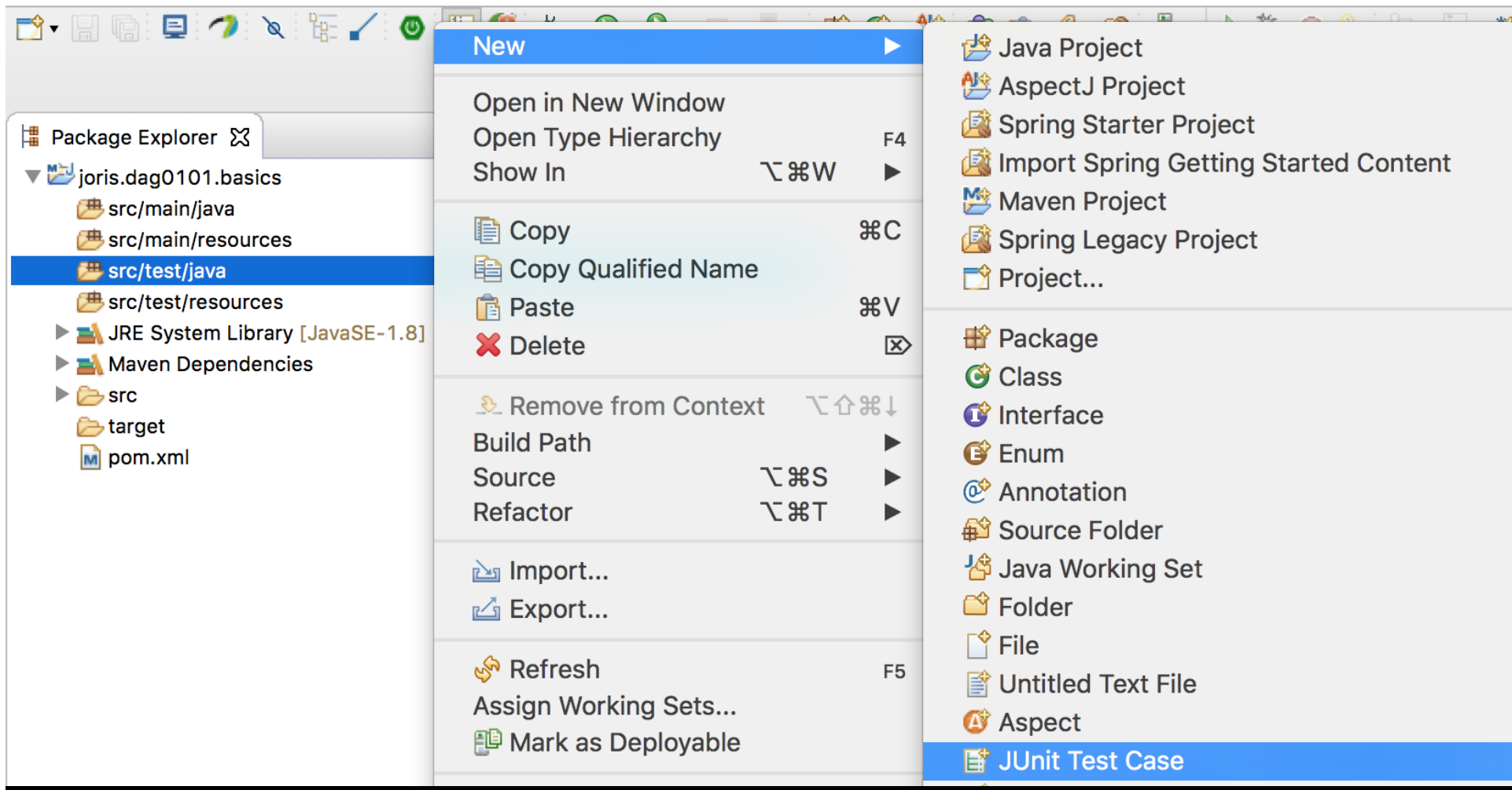
```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www
  <modelVersion>4.0.0</modelVersion>
  <groupId>nl.belastingdienst.courses.jft</groupId>
  <artifactId>joris.dag0101.basics</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
</project>
```


joris.dag0101.basics	Run As	Add Dependency
src/main/java	Debug As	Add Plugin
src/main/resources	Profile As	New Maven Module Project
src/test/java	Validate	Download JavaDoc
src/test/resources	Restore from Local History...	Download Sources
JRE System Library [J]	Maven	Update Project... F5
src	Team	Select Maven Profiles... ^ P
target	Compare With	Disable Workspace Resolution
pom.xml	Configure	Disable Maven Nature
	SonarLint	
	Spring Tools	

Add junit framework to the project

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

- When saving the maven client will fetch the junit framework



New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.



☐ New JUnit 3 test ☒ New JUnit 4 test

Source folder:

[Browse...](#)

Package:

[Browse...](#)

Name:

Superclass:

[Browse...](#)



[< Back](#)

[Next >](#)

[Cancel](#)

[Finish](#)

Package Explorer

▼ joris.dag0101.basics

src/main/java

src/main/resources

▼ src/test/java

▼ joris.dag0101.basics

▶ ExploreVariables.java

src/test/resources

▶ JRE System Library [JavaSE-1.8]

▶ Maven Dependencies

▶ src

target

pom.xml

Java code skelet

```
package joris.dag0101.basics;

import static org.junit.Assert.*;

import org.junit.Test;

public class ExploreVariables {

    @Test
    public void test() {
        //In here we will write java code
    }

}
```

Printing to the console

```
@Test
public void test() {
    //Inserte statement here
    System.out.println("Hello TestWorld!");
}
```

ExploreVariables.java

1 package joris.dag010

2

3 import static org.junit

6

7 public class Explore

8

9 @Test

10 public void test

11 System.out.p

12 }

13

14 }

15

Declarations

Add to Snippets...

AspectJ Refactoring

Run As

Debug As

Profile As

Validate

Team

Compare With

Replace With

Preferences...

ems Declaration Console

ables to display at this time.

1 Run on Server

2 JUnit Test

Run Configurations...



settings.xml - Kladblok

Bestand Bewerken Opmaak Beeld Help

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <localRepository>C:/ws/repo/${USERNAME}</localRepository>

  <servers>

</servers>

  <mirrors>
    <mirror>
      <id>belastingdienst.repo</id>
      <name>Repository belastingdienst</name>
      <url>http://rms.belastingdienst.nl/repo</url>
      <mirrorOf>*</mirrorOf>
    </mirror>
  </mirrors>
</settings>
```

Preferences

- + General
- + Ant
- + Code Recommenders
 - Gradle
- + Help
- + Install/Update
- + Java
- Maven
 - Archetypes

User Settings

Global Settings ([open file](#)):

User Settings ([open file](#)):

Hoe zouden jullie het concept variable beschrijven?

- * een doosje/container
- * is er voor een bepaalde tijd
- * doosjes specifiek voor bepaalde inhoud
- * kunt er iets instoppen/uithalen
- * kunt het hergebruiken

Typesystem Java

- 2 hoofdtakken
 - primitivetypes
 - referencetypes
- Kijken eerst naar primitivetypes -> 8 in totaal

Variabelen

- create new maven java project

```
package joris.jft.dag0102.primitives;

import static org.junit.Assert.*;

import org.junit.Test;

public class ExploreJavaPrimitives {

    @Test
    public void declarIntVariabeleEnAssignDeWaarde10() {
        //Schrijf hier code
    }
}
```

Variable kan te klein zijn voor een waarde

```
@Test
public void doosjeKunnenNietOnbeperktGroteGetallenBevatten() {
    // 4 bytes groot en 32 posities
    int doosje;

    // 2 forward slashes is een line comment
    // er is een max aan het getal dat we kunnen opslaan.
    doosje=1000000000;
    // dit lukt nog net
    doosje=Long.MAX_VALUE;
}
```

Primitive types:

- doosjes in java moeten getypeerd zijn
 - d.w.z. java wil van te voren weten wat je er in stopt
- we zullen allereerst stilstaan bij de primitive types
- later gaan we naar de reference types kijken

Kleinste type voor gehele getallen

- byte -> 8 bits

```
@Test
public void kleinsteMaatVoorGeheleGetallen() {
    byte doosjeVoor1Byte;
    doosjeVoor1Byte=(byte)(1 + 1);
    System.out.println(doosjeVoor1Byte);
}
```


Byte MAX_VALU

```
@Test
public void omslagPuntVanMaximaleGrootteNaarMinimaleGrootte()
{
    byte doosjeVoor1Byte=127;
    System.out.println("Maximale byte waarde = " + doosjeVoor1Byte);
    // + operator ziet 2 bytes-> telt tie netjes maar geeft een int +
    // int is 4 bytes past niet in een byte -> resultaat moet gecast
    doosjeVoor1Byte=(byte)(doosjeVoor1Byte + 1);
    System.out.println(doosjeVoor1Byte);
}
```

- Speciaal gedrag wanneer by de Byte.MAX_VALUE er 1 wordt opgeteld

short 2* grotere capaciteit dan byte

```
@Test
public void shortIs2KeerZoGrootAlsByte() {
    short shortDoosje=128;
    shortDoosje=(short) (shortDoosje+1);

    //Opmerkelijk genoeg gaat dit wel goed!
    shortDoosje += 1;

    System.out.println(shortDoosje);
}
```

long (8 bytes groot)

```
@Test
public void longTypeIs8BytesLang() {
    long longDoosje1=9_223_372_036_854_775_807L;
    //long longDoosje2=1l; kleine l niet gebruiken

    long maxLongDoosje=Long.MAX_VALUE;
    System.out.println(maxLongDoosje);
}
```

- long's kunnen groot worden, merk de _ (underscore) op als 1000-tal scheider

double type -> "approximate numerics"

- floatingpoint rekenen

```
@Test
public void doublesHebben8BytesOmGebrokenGetallenteBeschrijven() {
    double doubleDoosje1=0.2-0.1;
    double doubleDoosje2=0.3-0.2;
    System.out.println(doubleDoosje1-doubleDoosje2);
}
```

- Output in console: 2.7755575615628914E-17

Het float type

```
@Test
public void byDefaultWordenGebrokenGetallenGezienAlsDouble(){
    float floatDoosje=(float)1.0;
    System.out.println(floatDoosje);
}
```

- 1.0 zonder (float) wordt gezien als double -> 8 bytes -> past niet

Het boolean type

```
@Test
public void booleansExperiment() {
    boolean isDoosjeLeeg;
    isDoosjeLeeg=false;
    //Kijkt uit voor isDoosjeLeeg = isDoosjeLeeg -> dit is geen vergelijl
    isDoosjeLeeg=(isDoosjeLeeg == isDoosjeLeeg);
    System.out.println(isDoosjeLeeg);
}
```

- Boolean domein bevat 2 literal values: true en false

Het char type

```
@Test
public void spelenMetChars() {
    char charDoosje1='a';
    char charDoosje2='b';
    System.out.println(charDoosje1 + charDoosje2);

    System.out.println("dit is een waarde = " + 10);

    System.out.println(charDoosje1 + charDoosje2 + "");
}
```

- De + operator ziet char als getallen (unsigned)
- De expressie wordt van links -> rechts geevalueerd
- "" + 'a' -> String

Resumerend

- 8 primitive types
 - gehele getallen met sign
 1. byte 1 byte
 2. short 2 bytes
 3. int 4 bytes
 4. long 8 bytes
 - floating point
 1. float 4 bytes
 2. double 8 bytes
 - character
 1. char 2 bytes
 - boolean
 2. boolean

Basic syntax explored

```
@Test
public void hoeZietDeJavaSyntaxErEigenlijkUit() {
    // Met { .... } definieren we een blok
    System.out.println("Start van code blok");

    {
        //blocken kun je ook nesten
        System.out.println("Start van inner code blok");
        System.out.println("Einde van inner code blok");
    }
    System.out.println("Einde van code blok");
}
```

Scope van variabelen

```
@Test
public void hoeZitHetMetVariabelenEnCodeBlocken() {
    System.out.println("Start van code blok");
    boolean outerBlokBoolean=true;

    System.out.println("Outer block boolean = " + outerBlokBoolean);
    {
        System.out.println("Start van inner code blok");
        boolean innerBlokBoolean=true;

        System.out.println("Inner block boolean = " + innerBlokBoolean);
        System.out.println("Outer block boolean = " + outerBlokBoolean);
        System.out.println("Einde van inner code blok");
    }
    //System.out.println("Inner block boolean = " + innerBlokBoolean);
    System.out.println("Outer block boolean = " + outerBlokBoolean);
    System.out.println("Einde van code blok");
}
```

Scope van een variabele

- variable is zichtbaar binnen het block waarin het gedeclareerd wordt inclusief genested blocken.
- variabelen gedeclareerd in een genest block is er buiten niet zichtbaar.
- het block (inclusief geneste blocken) waarbinnen de variabele gebruikt kan worden wordt ook wel de scope van de variabele genoemd

Een blok wel of niet uitvoeren

```
@Test
public void hoeKunnenWeEenBlockConditioneelUitvoeren() {
    System.out.println("Start van code blok");

    boolean innerBlockUitvoeren=false;

    //Het geneste block willen we conditioneel uitvoeren
    //Syntax if
    // if(conditie){ statements} ->
    //als conditie true is dan worden statements
    if(innerBlockUitvoeren)
    {
        System.out.println("Start van inner code blok");
        System.out.println("Einde van inner code blok");
    }
    System.out.println("Einde van code blok");
}
```

Of het een of het andere

```
@Test
public void hoeKunnenHetEneBlockOfHetAndereBlockConditioneelUitvoeren() {
    System.out.println("Start van code blok");

    boolean uitvoerenBlok1=true;

    if(uitvoerenBlok1)
    {
        System.out.println("Start van inner code blok1");
        System.out.println("Einde van inner code blok1");
    }
    else
    {
        System.out.println("Start van inner code blok2");
        System.out.println("Einde van inner code blok2");
    }

    System.out.println("Einde van code blok");
}
```

Veel gebruikte syntaxconventie

```
@Test
public void erwordenVaakSyntaxConventiesGebruikt() {
    System.out.println("Start van code blok");

    boolean uitvoerenBlok1=true;

    if(uitvoerenBlok1){

        System.out.println("Start van inner code blok1");
        System.out.println("Einde van inner code blok1");

    }else{

        System.out.println("Start van inner code blok2");
        System.out.println("Einde van inner code blok2");

    }
}
```

Herhalen van een blok

```
@Test
public void hoeKunnenWeHetGenesteBlokEenAantalKerenHerhalen() {
    // Met { .... } definiëren we een blok
    System.out.println("Start van code blok");
    //Voorbeeld van oneindige loop
    while(true)
    {
        //blocken kun je ook nesten
        System.out.println("Start van inner code blok");

        System.out.println("Einde van inner code blok");
    }
    //System.out.println("Einde van code blok");
}
```


Vast aantal keer herhalen van blok

```
@Test
public void hoeKunnenWeHetGenesteBlok10KeerHerhalen() {
    int aantalHerhalingen=10;
    int loopNr=1;
    boolean conditie=true;
    while(conditie)
    {
        System.out.println("loopNr = " + loopNr + " " + "aantalHerhalingen=" + aantalHerhalingen);
        conditie=loopNr <= aantalHerhalingen;
        loopNr=loopNr+1;
    }
}
```

Restructure code in while statement

```
int aantalHerhalingen=10;
int loopNr=1;
boolean conditie=true;
while(conditie)
{
    // Doe useful work
    conditie=loopNr <= aantalHerhalingen;
    loopNr=loopNr+1;
} // restructure into

while(conditie=loopNr <= aantalHerhalingen)
{
    // Doe useful work
    loopNr=loopNr+1;
}
```

Translate while into for loop

```
int aantalHerhalingen=10;
int loopNr=1;
while(conditie=loopNr <= aantalHerhalingen)
{
    // Doe useful work
    loopNr=loopNr+1;
}
//translate into:
for(int loopNr=1;conditie=loopNr <= aantalHerhalingen;loopNr=loopNr+1;){
    // Doe useful work
}
```

Giving blocks a name

- Start with last example: calculation the table of 7

```
@Test
public void uitwerkingPrintenVanDeTafelVan7(){

    int finishedLoops = 0;
    int multiplyBy = 1;

    while (finishedLoops < 10) {
        System.out.println(multiplyBy + " * 7 = " +multiplyBy * 7);
        multiplyBy = multiplyBy + 1;
        finishedLoops = finishedLoops + 1;
    }
}
```

The table of 7

- What part of the code is really about the table of 7?

```
@Test
public void uitwerkingPrintenVanDeTafelVan7(){

    {
        int finishedLoops = 0;
        int multiplyBy = 1;

        while (finishedLoops < 10) {
            System.out.println(multiplyBy + " * 7 = " +multiplyBy * 7);
            multiplyBy = multiplyBy + 1;
            finishedLoops = finishedLoops + 1;
        }
    }

}
```

Gives block a descriptive name

```
@Test
public void uitwerkingPrintenVanDeTafelVan7() {

    tafelVan7() // <- When block receives a name () are necessary
    {
        int finishedLoops = 0;
        int multiplyBy = 1;

        while (finishedLoops < 10) {
            System.out.println(multiplyBy + " * 7 = " + multiplyBy * 7);
            multiplyBy = multiplyBy + 1;
            finishedLoops = finishedLoops + 1;
        }
    }
}
```

Rules about naming a block of code

- It is not allowed to give a block of code a name inside another block
- A block of code can supply a return value, it can give back a value
- In Java you must specify the type a block can return
- If the block does not return anything you should use void as a return type.

The code

- tafelVan7() is called a method

```
void tafelVan7(){
    int finishedLoops = 0;
    int multiplyBy = 1;

    while (finishedLoops < 10) {
        System.out.println(multiplyBy + " * 7 = " + multiplyBy * 7);

        multiplyBy = multiplyBy + 1;
        finishedLoops = finishedLoops + 1;
    }
}
```


Calling a method

- In the code the codeblock is executed twice

```
@Test
public void generalisatie(){

    // block code geven we een naam
    tafelVan7();
    tafelVan7();

}
```

Een methode generieker maken

```
void tafelVan7Opgeschoond(){  
    int finishedLoops = 0;  
    int multiplyBy = 1;  
  
    while (finishedLoops < 10) {  
        p(multiplyBy + " * 7 = " + multiplyBy * 7);  
  
        multiplyBy = multiplyBy + 1;  
        finishedLoops = finishedLoops + 1;  
    }  
}
```

Introductie parameter

```
void tafelVan(int grondTal) {  
    int finishedLoops = 0;  
    int multiplyBy = 1;  
  
    while (finishedLoops < 10) {  
        p(multiplyBy + " * " + grondTal + " = " + multiplyBy * grondTal);  
        multiplyBy = multiplyBy + 1;  
        finishedLoops = finishedLoops + 1;  
    }  
}
```

Aanroepen van het code blok met parameter

```
@Test  
public void voerDeTafelVan7EnVan8Uit(){  
    tafelVan(7);  
    tafelVan(8);  
}
```

System.out.println geeft veel typewerk

```
@Test
public void getRidOfSystemoutprintln(){
    // Hier hebben we wel een block code uitgehaald
    // Hier roepen we de methode aan

    p();

    p("nog een andere string");

    // methods with the same name but with different
    // number of parameters or type of parameter are allowed in Java
    // This is called method overloading
}
```

Methods with the same name

```
// declareren we een methode
void p(){
    System.out.println("Tekst");
}
//String input wordt parameter genoemd
void p(String input){
    System.out.println(input);
}
```

Methode overloading

- methodes in java mogen de zelfde naam hebben
- methodes moeten dan wel verschillen in
- aantal parameters
- of type van parameters

Het zelf maken van objecten

```
@Test
public void hetAanmakenVanRoald() {
    Persoon roald; // maak een doosje -> reference persoon object
}
```

- We moeten een object maken van het type Persoon
- Het type Persoon bestaat nog niet
- In java is echter class synoniem voor bouwtekening
- We kunnen zelf een class maken met de naam Person

De class Persoon

```
public class Persoon{  
  
}
```

- De Persoon class is een leeg omhulsel
- Er is nog niet beschreven wat een object van dit type moet
 1. weten
 2. doen
- We kunnen al wel een (bijna kaal) object instantiëren.

```
@Test
public void hetAanmakenVanRoald() {

    Persoon roald;
    // maak een doosje -> mag een reference naar persoon object in

    roald = new Persoon();
    // de new operator -> maakt een nieuwe persoon object aan

}
```

class als bouwtekening

- Wat moet de class kunnen -> opnemen in bouwtekening

```
@Test
public void hetAanmakenVanRoaldEnFerwin() {
    Persoon persoon1 = new Persoon();

    persoon1.setNaam("Roald");

    persoon1.setLeeftijd(52);

    String naam=persoon1.getNaam();

    int leeftijd=persoon1.getLeeftijd();

    System.out.println("Persoon met naam " + naam + " met leeftijd " + leeftijd);
}
```

```
@Test public void hetAanmakenVanRoaldEnFerwin() { Persoon  
    persoon1 = new Persoon();
```

```
    persoon1.setNaam("Roald");
```

```
    persoon1.setLeeftijd(52);
```

```
    String naam=persoon1.getNaam();
```

```
    int leeftijd=persoon1.getLeeftijd();
```

```
    System.out.println("Persoon met naam " + naam + " met leeftijd " + leeft:
```

```
    Persoon persoon2 = new Persoon();
```

```
    persoon2.setNaam("Ferwin");
```

```
    persoon2.setLeeftijd(52);
```

```
    String naam2 = persoon2.getNaam();
```

```
}
```

//Nabouwen van bovenstaande

```
@Test public void watKunnenWeNuMetReferenceVariabelen() {  
    //8 primitivetypes -> the value is stored inside the variabel  
    byte byteDoosje=1; short shortDoosje=32000; int  
    intDoosje=2_000_000; long longDoosje=1_000_000_000L; float  
    floatDoosje=1.0F; double doubleDoosje=1.0; boolean  
    trueFalseDoosje=true; char charDoosje='a';
```

```
//The Reference types
```

```
Persoon persoon1 = new Persoon();
```

```
System.out.println("Naam = " + persoon1.getNaam() + " en leeftijd = " + }  
}
```

```
@Test public void  
eenObjectDatAangemaaktIsBevindZichAltijdInWelgedefineerdeToes  
{
```

```

Persoon persoon1 = new Persoon();

System.out.println("Leeftijd = " + persoon1.getLeeftijd() + " en humor = " + persoon1.getHumor());

//-> primitive instance variabelen worden op 0 of 0.0 of false geïnitieerd.

System.out.println("De naam = " + persoon1.getNaam());

// -> Een Reference instancevariabele die geen waarde krijgt wordt op null gezet.

}

```

@Test public void watGebeurtErHier() { Persoon persoon2= new Persoon(); persoon2.setLeeftijd(56); persoon2.setNaam("Roald");

```

Persoon diePeter= new Persoon();
diePeter.setNaam("Peter");
diePeter.setLeeftijd(49);

```

```

Persoon anderePeter= new Persoon();
anderePeter.setNaam("Peter");
anderePeter.setLeeftijd(53);

```

//We zijn hier bij het uitvoeren van de code

```

System.out.println("De naam = " + persoon2.getNaam() + "Leeftijd = " + persoon2.getLeeftijd());
System.out.println("De naam = " + diePeter.getNaam() + "Leeftijd = " + diePeter.getLeeftijd());
System.out.println("De naam = " + anderePeter.getNaam() + "Leeftijd = " + anderePeter.getLeeftijd());

```

```
}
```

```
@Test public void enWatGebeurtErHier() {
```

```
Persoon persoon2= new Persoon("Roald",56);
```

```
Persoon diePeter= new Persoon("Peter",49);
```

```
Persoon anderePeter= new Persoon("Peter",53);
```

```
//We zijn hier bij het uitvoeren van de code
```

```
System.out.println("De naam = " + persoon2.getNaam() + "Leeftijd = " + pe
```

```
System.out.println("De naam = " + diePeter.getNaam() + "Leeftijd = " + d
```

```
System.out.println("De naam = " + anderePeter.getNaam() + "Leeftijd = " -
```

```
}
```

```
@Test public void deToestandVeranderenInEenObject() { Persoon  
    persoon = new Persoon("Roald",-56);
```

```
//persoon.humor=true;
```

```
System.out.println("De naam = " + persoon.getNaam() + "Leeftijd = " + pe
```

}