Aandachtspunten

Code Stijl:

- Gebruik intentie onthullende namen.

Goed:
```
public void print(String str) {
        System.out.println(str);
}
```

Slecht:
```
public void banaan(String str) {
        System.out.println(str);
}
```

- Klasse namen beginnen met een hoofdletter, methode namen beginnen met een kleine letter. Beide mogen in camelCase.

Goed:
```
public Class Main {
        public void getName() {
                return name;
        }
}
```

Slecht:
```
public Class main {
        public void Getname() {
                return name;
        }
}
```

- Geen uitgecommentarieerde code laten staan.

Goed: // Opdracht 1

public void first () {

...code...

}

Slecht: // Opdracht 1

// public void second() {

// …code...

// }

public void first() {

..code..

}

- Je xml code gebruikt correcte inspring.

Goed: <dependencies>

<dependency>

..code..

</dependency>

</dependencies>

Fout: <dependencies><dependency>

...code..

</dependency><dependencies>

- Methodes doen maar één ding.

Goed: public String getName(){

return name;

}

Fout: public String getName(int age){

this.age =age;

return name;

}

Functionaliteit

- Er zitten 2 klassen in het project (Person en Pet)
- Person heeft 10 attributen en 2 constructors
- Pet heeft 4 attributen en 1 constructor
- Alle attributen hebben een getter en een setter
- Person bevat nog de methodes: addParents, aadChild, addPet, addSibbling en getGrandChildren.
- Het project bevat de Maven mappen structuur:
    - Src
        - Main
            - Java (blauw/ sources root)
            - Resources
        - Test
            - Java (groen/ test root)
            - Resources
- Er staat een pom.xml bestand in de root folder van het project
- In de <version>-tag van de pom staat geen “SNAPSHOT”.
- In de pom is junit 5 (jupiter) toegevoegd als dependency
- Elke test is geannoteerd met “@Test”
- Elke test is opgebouwd volgens de Arrange/Act/Assert methodiek.
- Elke methode wordt getest, maar getters en setters mogen wel in één test samen getest worden.