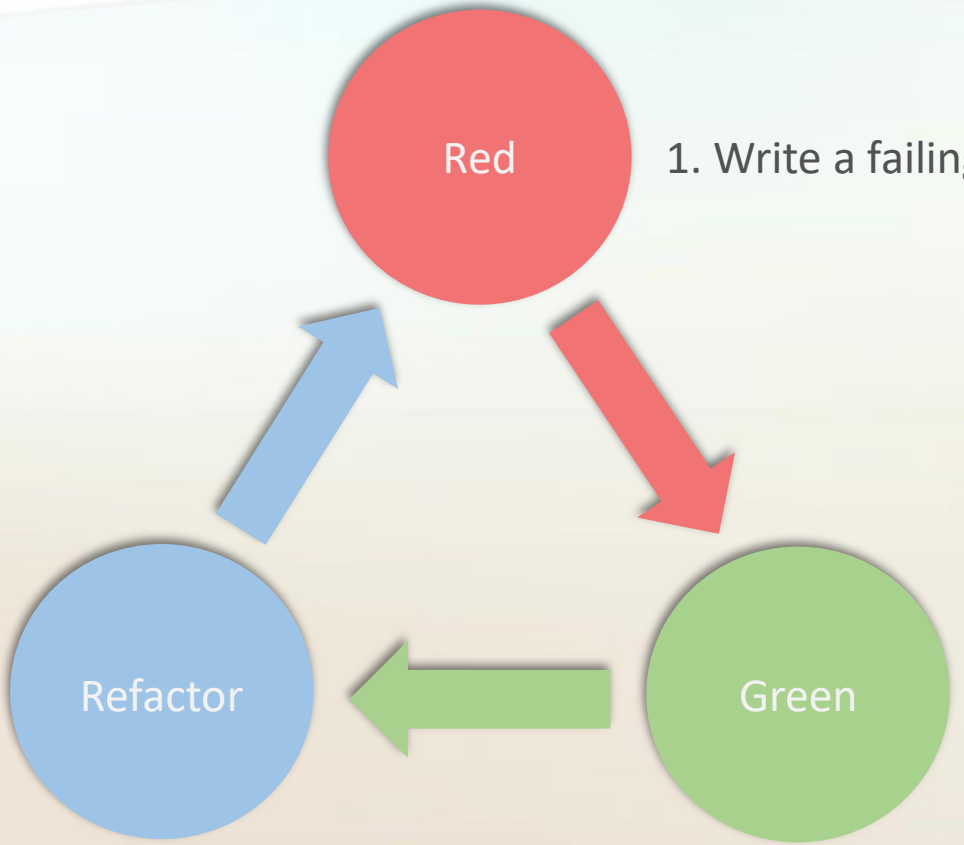
# Test Driven Development – TDD

(Test Driven Design…)

sogeti
Part of Capgemini

# What is Test Driven Development?




1. Write a failing test

2. Make the test pass

3. Optimize the code's design

# Starting with the test

```
public void AddCourse(Course course);
```

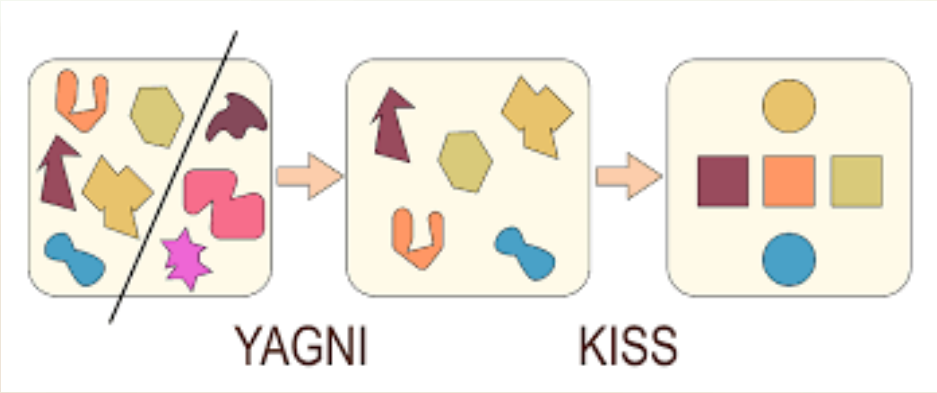
# Starting with the test

```
public void AddCourse(Course course)
{
      var errors = course.IsValid();

       If(errors.Any())
       {
              _repository.AddValidationErrors(errors);
       }
       else
       {
              _repository.Add(course);
       }
}
```

# Having a failing test

- Build exactly what is needed – nothing more!
- Unnecessary features – big reason for system failures.
- YAGNI – You Aren't Gonna Need It
- KISS – Keep It Simple, Stupid!



# Make it Pass!

```
public void IsApprovedForCSN_WhenAgeIsOver56_ReturnsFalse()
{
        //Arrange
        var student = new Student();
        student.Age = 57;
        var validator = new StudentValidator(student);

        //Act
        var result = validator.IsApprovedForCSN();

        //Assert
        Assert.False(result);
}
```

# Make it Pass!

```
public void IsApprovedForCSN_WhenAgeIsOver56_ReturnsFalse()
{
        //Arrange
        var student = new Student();
        student.Age = 57;
        var validator = new StudentValidator(student);

        //Act
        var result = validator.IsApprovedForCSN();

        //Assert
        Assert.False(result);
}
```

```
public bool IsApprovedForCSN()
{
        return false;
}
```

# Make it Pass!

```
public void IsApprovedForCSN_WhenAgeIsUnder57_ReturnsTrue()
{
        //Arrange
        var student = new Student();
        student.Age = 30;
        var validator = new StudentValidator(student);

        //Act
        var result = validator.IsApprovedForCSN();

        //Assert
        Assert.True(result);
}
```

# Make it Pass!

```
public void IsApprovedForCSN_WhenAgeIsUnder57_ReturnsTrue()
{
        //Arrange
        var student = new Student();
        student.Age = 30;
        var validator = new StudentValidator(student);

        //Act
        var result = validator.IsApprovedForCSN();

        //Assert
        Assert.True(result);
}
```

```
public bool IsApprovedForCSN()
{
        return _student.Age  < 57;
}
```

# Refactoring

- Change the code without changing its behavior.
- For example:
  - Breaking out a part of code from one method into a new method.
  - Breaking out new classes from one big class.
- Remove duplicated code.
- Duplicated code becomes a problem when changes occur.

# Word of advice

- **Best when:**
  - Expected behavior is known.
  - Lots of use cases/ scenarios.
  - Team has a similar drive to use it.
  - Product owner has knowledge about costs.

- **More difficult to use when:**
  - Requirements are not really known.
  - Experimenting with code design.
  - Developer is inexperienced.

# This presentation can be found at:

- https://github.com/starefeldt/presentation-unittest-tdd

# Resources:

- *Test Driven Development, By Example* – Kent Beck (2003)
- *Working Effectively with Legacy Code* – Michael Feathers (2004)
- *Clean Code* – Robert C Martin (2008)
- *The Art of Unit Testing* – Roy Osherove (2014)
- *Dependency Injection Principles, Practices, and Patterns* – Steven van Deursen and Mark Seemann (2019)
- *Pluralsight* – online learning platform