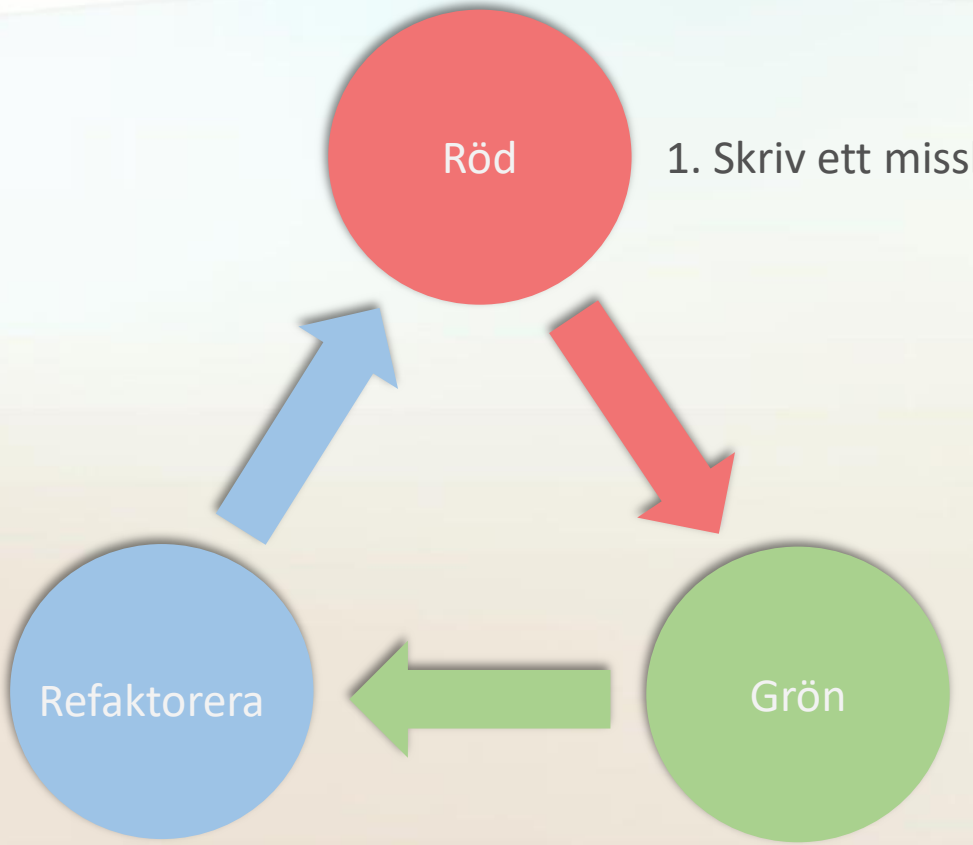
Testdriven utveckling– TDD

(Testdriven design…)

Vad är Testdriven utveckling?



1. Skriv ett misslyckat test

2. Få testet att passera

3. Optimera kodens design

Börja med testet

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

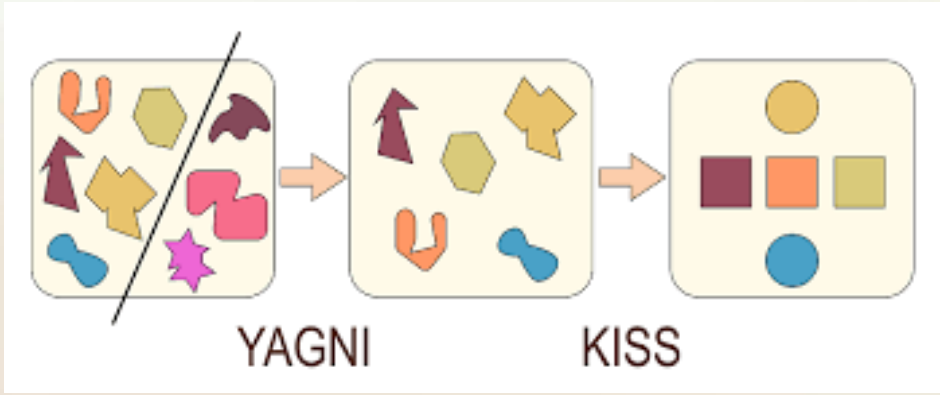
Börja med testet

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

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

Ett misslyckat test

- Bygg exakt det som behövs – inget mer!
- Onödiga funktioner – stor anledning till dåliga system
- YAGNI – You Aren’t Gonna Need It
- KISS – Keep It Simple, Stupid



Passera testet!

```
public void IsApproved_WhenAgeIsUnder18_ReturnsFalse()
{
        //Arrange
        var student = new Student();
        student.Age = 17;
        var validator = new StudentValidator(student);

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

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

Passera testet!

```
public void IsApproved_WhenAgeIsUnder18_ReturnsFalse()
{
        //Arrange
        var student = new Student();
        student.Age = 17;
        var validator = new StudentValidator(student);

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

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

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

Passera testet!

```
public void IsApproved_WhenAgeIsOver17_ReturnsTrue()
{
        //Arrange
        var student = new Student();
        student.Age = 18;
        var validator = new StudentValidator(student);

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

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

Passera testet!

```
public void IsApproved_WhenAgeIsOver17_ReturnsTrue()
{
        //Arrange
        var student = new Student();
        student.Age = 18;
        var validator = new StudentValidator(student);

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

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

```
public bool IsApproved()
{
    return _student.Age  > 17;
}
```

Refaktorering

- Ändra koden utan att ändra beteendet
- Exampel:
  - Bryt ut en del av koden från en metod till en ny metod
  - Bryt ut nya klasser från en stor klass
- Ta bort duplicerad kod
- Duplicerad kod blir ett problem när ändringar genomförs

Goda råd

- **Bäst när:**
  - Förväntat beteende är känt
  - Mängder av use cases/ scenarion
  - Hela teamet har samma motivation att använda det
  - Produktägaren har kännedom om kostnader

- **Mer problematiskt när:**
  - Krav inte är kända
  - Experimenterande med kodens design
  - Utvecklaren är oerfaren och tidsbrist råder i projektet

Presentationen kan hittas här:

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

Referensmaterial:

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