Software Engineering Essentials

Dependency Injection

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics

From State Testing to Behavior Testing

- Observation: Mock objects help to test behavior
- Limitation of mock objects:
  - Mock objects might lead to high coupling between SUT and the rest of the system model

We would like to reduce this coupling as much as possible

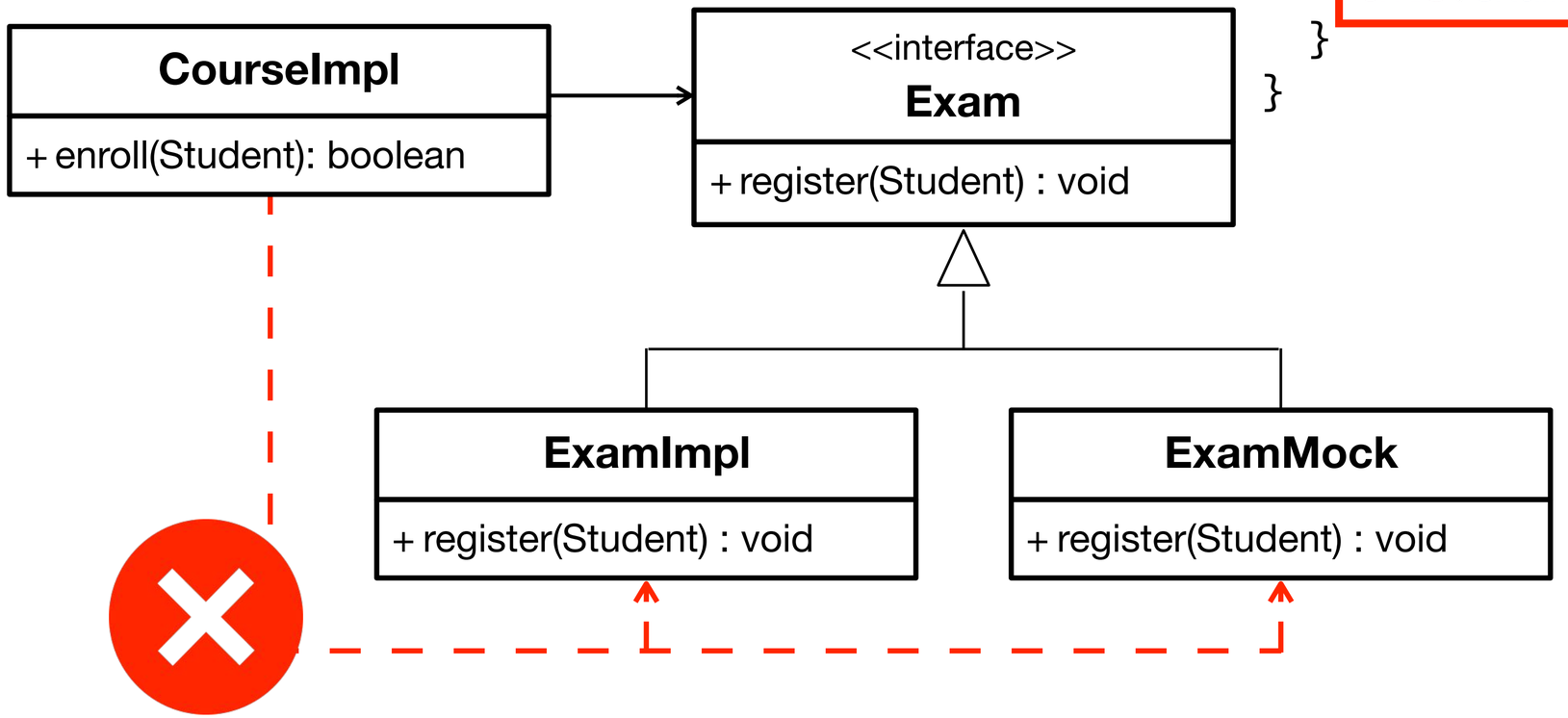
- Dependency injections comes into play

Learning Goals:

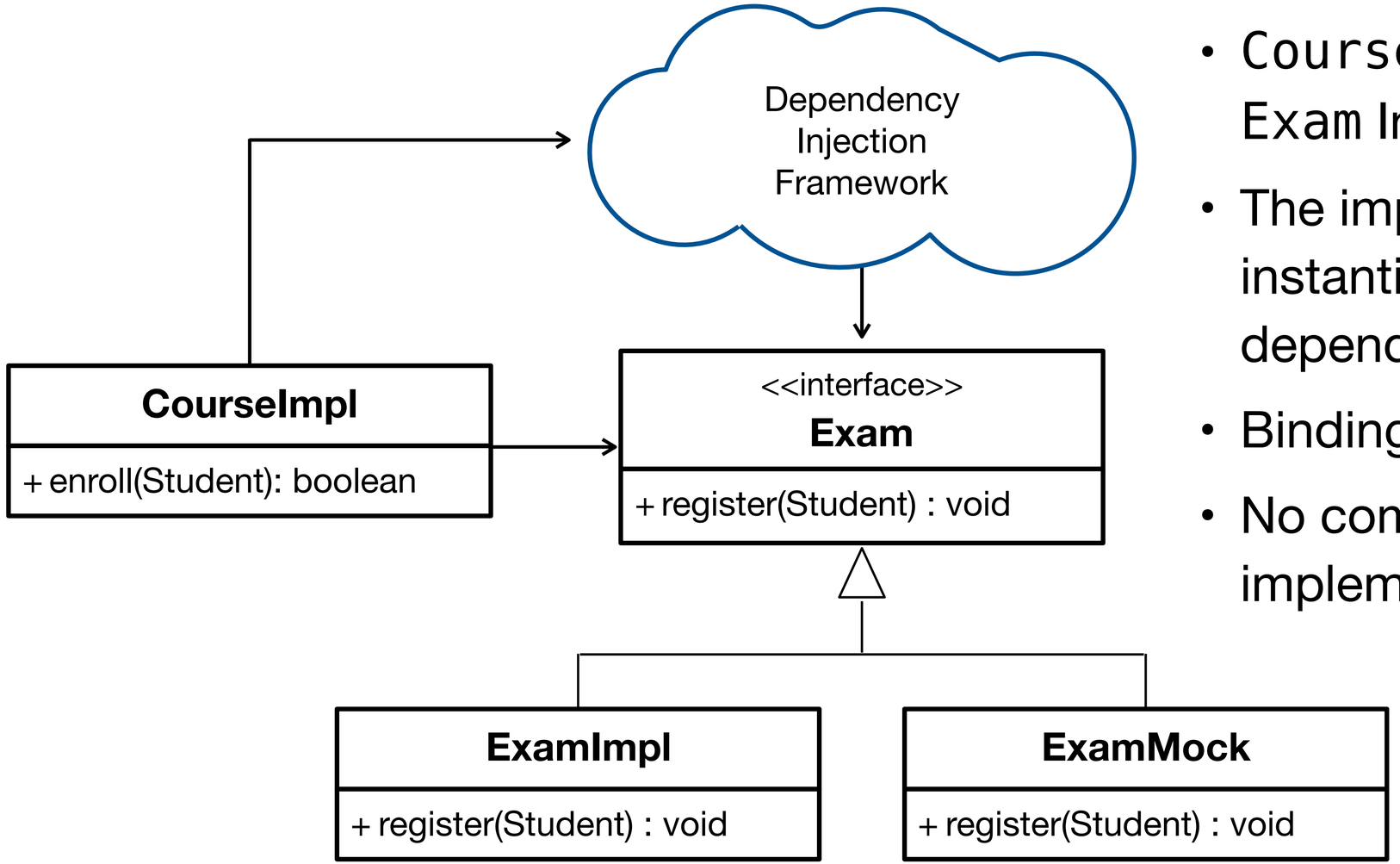
- Understand the concept of dependency injection

Problem High Coupling

```
public class CourseImpl implements Course
{

  private Exam exam;

  public CourseImpl() {
    this.exam = new ExamImpl();
  }
}
```



Problem High Coupling during Testing



- `CourseImpl` only knows the `Exam` Interface
- The implementations are instantiated and injected by a dependency injection framework
- Binding is done by the framework
- No compile-time dependencies to implementations

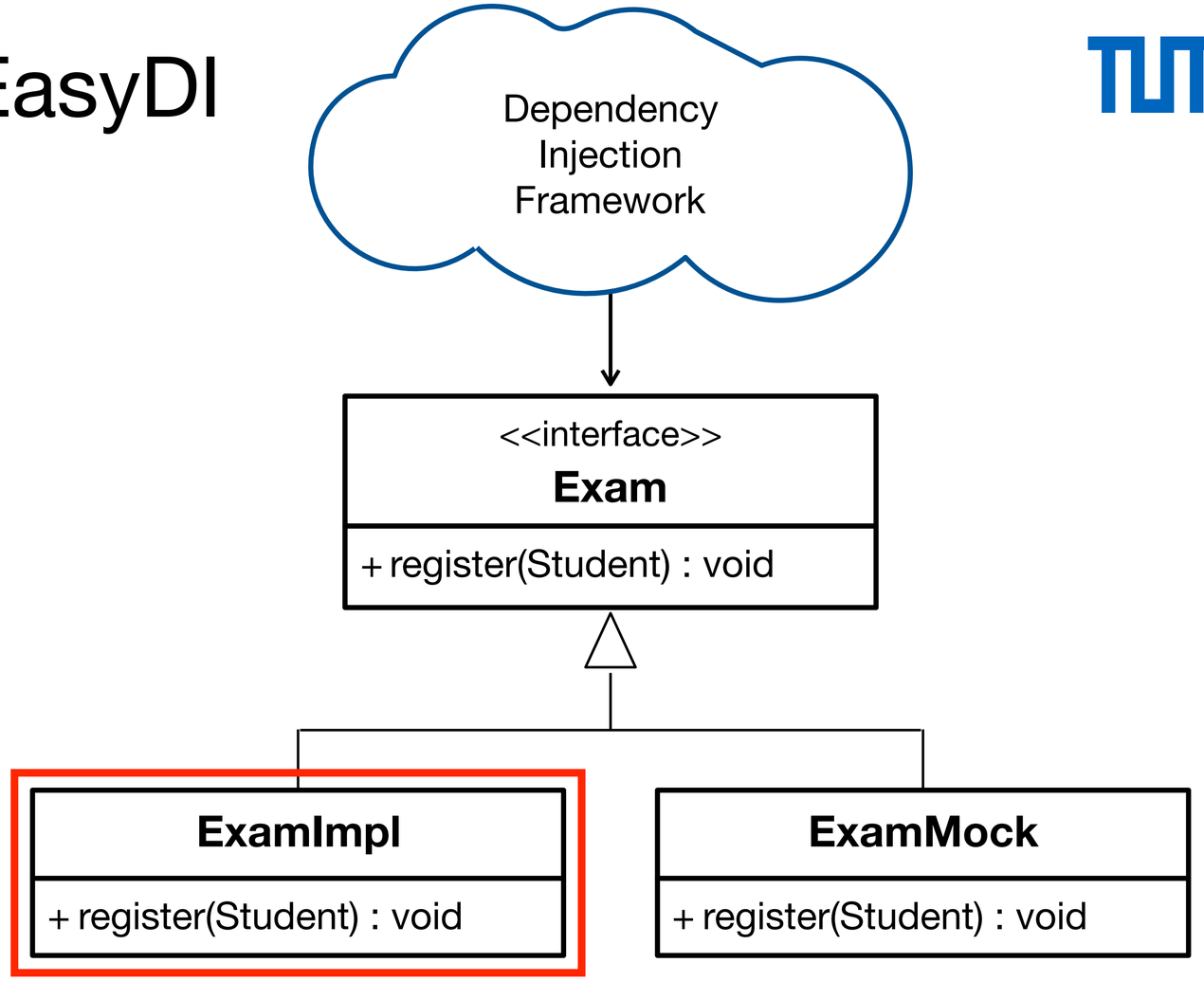
Dependency Injection with EasyDI

```
public class CourseImpl implements Course {

   private Exam exam;

   public CourseImpl() {
      this.exam = new ExamImpl();
   }
}
```

```
public class CourseImpl implements Course {

   private Exam exam;

   public CourseImpl() {
      EasyDI easyDI = new EasyDI();
      easyDI.bindInterface(Exam.class, ExamImpl.class);
      this.exam = easyDI.getInstance(Exam.class);
   }
}
```

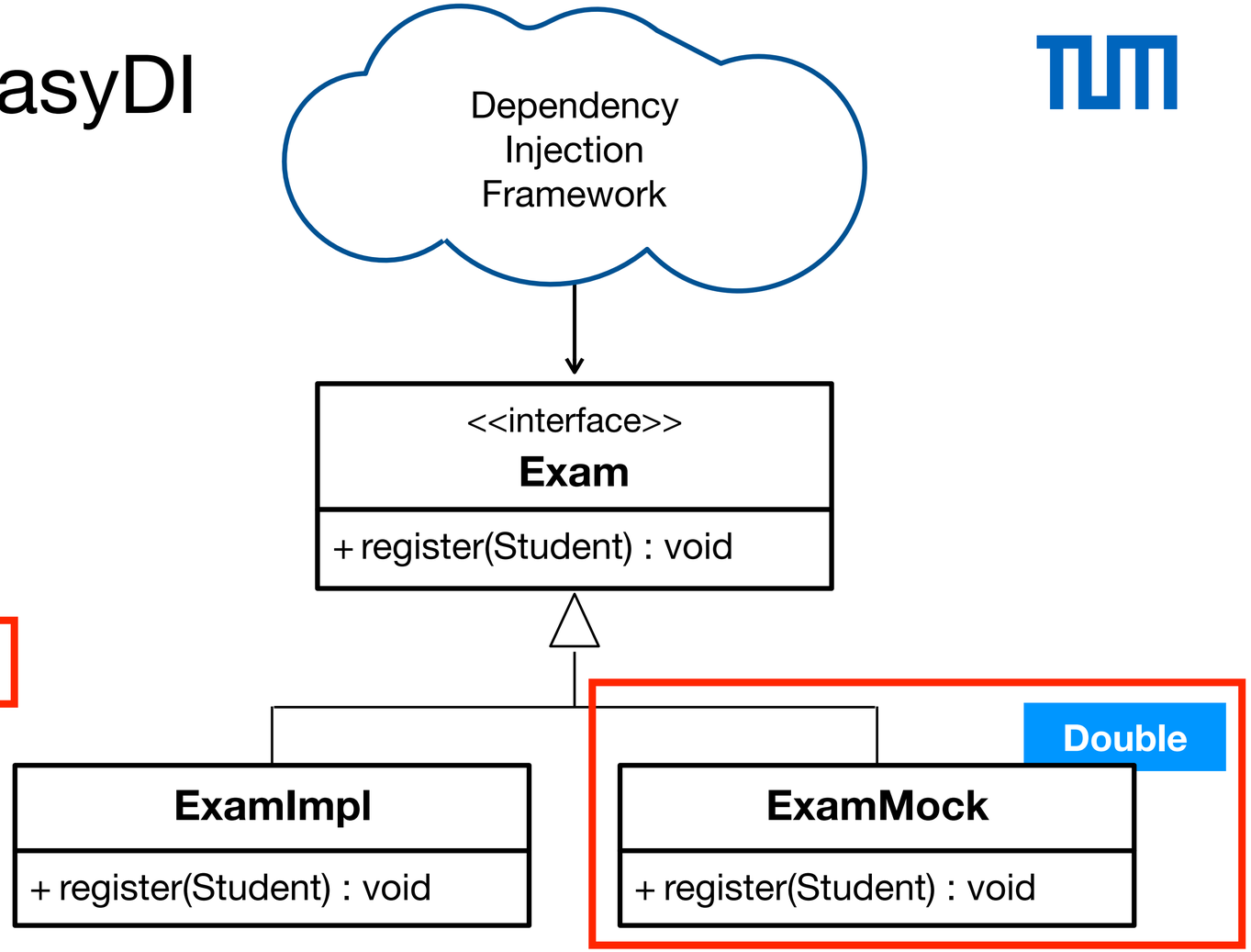
Implementation

Interface



Dependency Injection with EasyDI

```java
@RunWith(EasyMockRunner.class)
public class CourseImplTest {

   @TestSubject
   private CourseImpl course = new CourseImpl();

   @Mock
   private Exam examMock;

   @Test
   public void testCourse() {
      EasyDI easyDI = new EasyDI();
      easyDI.bindInterface(Exam.class, examMock.getClass());
      Student student = new Student("Andreas", "Seitz");

      expect(examMock.register(student)).andReturn(true);
      replay(examMock);

      course.enroll(student);

      verify(examMock);
   }

}
```



Software Engineering Essentials

Dependency Injection

Bernd Bruegge, Stephan Krusche, Andreas Seitz, Jan Knobloch
Chair for Applied Software Engineering — Faculty of Informatics