**Form Diskusi Pertemuan 13**

**OBJECT ORIENTED TESTING**

**Tujuan diskusi:**

* Mahasiswa mampu mengerti tentang teknik pengujian berorientasi objek.
* Mahasiswa mampu melakukan unit testing menggunakan Junit.

**Diskusi 01:**

|  |
| --- |
| 1. Dengan menggunakan Netbeans, buatlah project baru dengan nama **JUnitSample**. 2. Buatlah sebuah kelas dengan nama **Concat**, kelas ini adalah kode program yang akan diuji. Tuliskan kode program seperti berikut ini:   public class Concat {  public Concat() {}    public String concatenate(String firstName, String lastName)  {  return firstName + " " + lastName;  }  }   1. Pada project **JUnitSample** yang baru di-create, hanya ada 2 packages yaitu Source Packages dan Libraries, sehingga perlu menciptakan packages dan kelas untuk keperluan Testing, dengan cara: klik **Tools 🡪 Create/Update Tests**   Sehingga struktur project nya akan menjadi:     1. Periksa pada folder **Test Libraries**, apakah sudah terdapat library **JUnit** dan **Hamcrest**. Jika belum ada, dapat ditambahkan dengan 2 cara berikut ini (lakukan salah 1 saja): 2. Add Library   Klik kanan pada folder **Test Libraries**, pilih **Add Library**, pilih library **JUnit 4.13.2** dan **Hamcrest 1.3**  Sehingga menjadi:   1. Klik kanan pada folder **Test Libraries**, pilih **Add JAR/Folder**, cari library **JUnit** dengan ekstensi **.jar** yang tersimpan pada komputer Anda**.**   Sehingga menjadi:   1. Menulis unit test untuk menguji fungsi Concatenate. Pada folder **Test Packages**, terdapat sebuah file **ConcatTest.java** yang merupakan kelas pengujian untuk menuliskan kode unit test untuk menguji kelas Concat.   Tuliskan kode pengujian seperti berikut ini:  package junitsample;  public class ConcatTest extends junit.framework.TestCase {  public ConcatTest() {  }    public void testConcatenate() {  Concat C= new Concat();  String result=C.concatenate("Christian","Vieri");  assertEquals("Christian Vieri",result);  }  }   1. Menjalankan unit test dapat dilakukan dengan 2 cara seperti berikut: 2. Klik kanan pada file kode sumber yang diuji (file Concat.java), dan pilih **Test File**.  1. Klik kanan pada file pengujian (file ConcatTest.java), dan pilih **Run File**.   Pastikan hasilnya passed seperti ini, dan muncul detail pengujiannya juga. Jika belum muncul detialnya, klik tanda centang hijau : |
| 1. Masih pada project **JUnitSample**. Buatlah sebuah kelas dengan nama **Counter**, kelas ini adalah kode program yang akan diuji. Tuliskan kode program seperti berikut ini:   public class Counter {  int count = 0;  public Counter() {}    public int increment() {  return ++count;  }  public int decrement() {  return --count;  }  }   1. Menciptakan kelas **CounterTest.java** untuk keperluan Testing, dengan cara: klik **Tools 🡪 Create/Update Tests** dan periksa bahwa namanya adalah benar **CounterTest.** 2. Menulis unit test untuk menguji method-method pada kelas Counter. Pada folder **Test Packages**, terdapat sebuah file **CounterTest.java** yang merupakan kelas pengujian untuk menuliskan kode unit test untuk menguji kelas Counter.   Tuliskan kode pengujian seperti berikut ini:  package junitsample;  public class CounterTest extends junit.framework.TestCase  {  Counter counter1;  public CounterTest() {  }    protected void setUp() {  counter1 = new Counter();  }    public void testIncrement() {  assertTrue(counter1.increment() == 1);  assertTrue(counter1.increment() == 2);  assertTrue(counter1.increment() == 3);  }    public void testIncrementFalse() {  assertFalse(counter1.increment() == 0);  assertFalse(counter1.increment() == 1);  }    public void testDecrement() {  assertTrue(counter1.decrement() == -1);  }  }   1. Jalankan unit test **CounterTest** dan amati hasilnya! 2. Ubahlah kode pengujian **CounterTest** supaya test case **testIncrement** dan **testIncrementFalse** menghasilkan hasil pengujian yang failed. Screenshot hasil modifikasi kode pengujian yang Anda buat! Screenshot hasil beserta detail pengujiannya! Jelaskan mengapa menampilkan hasil pengujian tersebut? |
|  |
| 1. Masih pada project **JUnitSample**. Buatlah sebuah kelas dengan nama **Employee**, kelas ini adalah kode program yang akan diuji. Tuliskan kode program seperti berikut ini:   public class Employee {  private String Name;  private int Id;    public Employee() {  }    public Employee(String Name, int Id) {  this.Name=Name;  this.Id=Id;  }  }   1. Menciptakan kelas **EmployeeTest.java** untuk keperluan Testing, dengan cara: klik **Tools 🡪 Create/Update Tests** dan periksa bahwa namanya adalah benar **EmployeeTest.** 2. Menulis unit test untuk menguji kelas Employee. Pada folder **Test Packages**, terdapat sebuah file **EmployeeTest.java** yang merupakan kelas pengujian untuk menuliskan kode unit test untuk menguji kelas Employee.   Tuliskan kode pengujian seperti berikut ini:  package junitsample;  public class EmployeeTest extends junit.framework.TestCase {    public EmployeeTest() {  }    public void testEmployee(){  Employee E1= new Employee("Andi",45);  Employee E2= new Employee("Andi",45);  assertSame(E1,E2);  }    public void testEmployee2(){  Employee E1= new Employee("Andi",45);  assertNotNull(E1);  }  }   1. Jalankan unit test **EmployeeTest** dan amati hasilnya! 2. Jelaskan mengapa hasil pengujian tersebut failed untuk test case testEmployee dan passed untuk test case testEmployee2 ? |
|  |

**Diskusi 02:**

|  |
| --- |
| 1. Dengan menggunakan Netbeans, buatlah project baru dengan nama **Stack**. 2. Buatlah sebuah kelas dengan nama **SimpleStack01**, kelas ini adalah kode program yang akan diuji. Tuliskan kode program seperti berikut ini:   public class SimpleStack01 {  private int[] arr = null;  private int MAX = 5;  private int top = -1;  private int size = 0;    public SimpleStack01 () {  this.arr = new int[MAX];  }  public boolean isFull() {  return (size == MAX); //top==MAX-1  }    public boolean isEmpty() {  return (size == 0); //top==-1  }    public int getSize() {  return size;  }  public String push(int e) {  if (isFull())  throw new IllegalArgumentException("Stack Overflow");    size++;  arr[++top] = e;  return "Element " + e + " inserted succesfully";  }    public int pop() {  int result;  if(isEmpty()){  throw new java.util.NoSuchElementException("Stack Underflow");  }  size--;  result = arr[top];  this.top--;  return result;  }  }   1. Periksa pada folder **Test Libraries**, apakah sudah terdapat library **JUnit** dan **Hamcrest**. Jika belum ada, tambahkan kedua library ini. 2. Menciptakan kelas **PushTest.java** untuk keperluan Testing, dengan cara: klik **Tools 🡪 Create/Update Tests** dan periksa bahwa namanya adalah benar **PushTest.** 3. Menulis unit test untuk menguji method **Push** pada kelas **Stack**. Pada folder **Test Packages**, terdapat sebuah file **PushTest.java** yang merupakan kelas pengujian untuk menuliskan kode unit test untuk menguji method **push** pada kelas **Stack**.   Test case yang dibuat sesuai dengan yang sudah kita susun sebelumnya:  size<max  size=max  Tuliskan kode pengujian seperti berikut ini:  package stack;  public class PushTest extends junit.framework.TestCase {  SimpleStack01 instance;  public PushTest() {  }  public void setUp() {  instance = new SimpleStack01();  instance.push(1);  instance.push(2);  instance.push(3);  instance.push(4);  }  public void tearDown() {  }  public void testPushPass() {  System.out.println("push");  int e = 23;  String expResult = "Element " + e + " inserted succesfully";  String result = instance.push(e);  assertEquals(expResult, result);  }    public void testPushFail() {  System.out.println("push");  int e = 34;  String expResult = "Element " + e + " inserted succesfully";  String result = instance.push(e);  assertEquals(expResult, result);  result = instance.push(e);  assertEquals(expResult, result);  }  }   1. Menciptakan kelas **PopTest.java** untuk keperluan Testing, dengan cara: klik kanan pada package **Stack** di folder **Test Packages**, buatlah Java Class baru, dan beri nama **PopTest.**  1. Menulis unit test untuk menguji method **Pop** pada kelas **Stack**. Pada folder **Test Packages**, terdapat sebuah file **PopTest.java** yang merupakan kelas pengujian untuk menuliskan kode unit test untuk menguji method **pop** pada kelas **Stack**.   Test case yang dibuat sesuai dengan yang sudah kita susun sebelumnya:    top>-1  top=-1  size>0  size=0  Tuliskan kode pengujian seperti berikut ini:  package stack;  public class PopTest extends junit.framework.TestCase {  SimpleStack01 instance;  public PopTest() {  }    public void setUp() {  instance = new SimpleStack01();  instance.push(23);  }  public void tearDown() {  }  public void testPopPass() {  System.out.println("pop");  int expResult = 23;  int result = instance.pop();  assertEquals(expResult, result);  }    public void testPopFail() {  System.out.println("pop");  int expResult = 23;  int result = instance.pop();  assertEquals(expResult, result);  result = instance.pop();  assertEquals(expResult, result);  }  }   1. Menciptakan kelas **SimpleStack01TestSuite.java** sebagai Test Suite untuk menjalankan pengujian unit test **PushTest** dan **PopTest**, dengan cara: klik kanan pada package **Stack** di folder **Test Packages**, buatlah Java Class baru, dan beri nama **SimpleStack01TestSuite.**  1. Menulis kode pengujian pada **SimpleStack01TestSuite.java** seperti berikut ini:   package stack;  import org.junit.runner.RunWith;  import org.junit.runners.Suite;  @RunWith(Suite.class)  @Suite.SuiteClasses({stack.PushTest.class, stack.PopTest.class})  public class SimpleStack01TestSuite {    }   1. Jalankan test suite **SimpleStack01TestSuite** dan amati hasilnya! 2. Jelaskan mengapa hasil pengujian tersebut passed untuk test case testPushPass dan failed untuk test case testPushFail? |
|  |
| 1. Jelaskan mengapa hasil pengujian tersebut passed untuk test case testPopPass dan failed untuk test case testPopFail? |
|  |

**No Kelompok :**

**Anggota :**

|  |  |
| --- | --- |
| **NPM Anggota** | **Nama Anggota** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |