Puhas kood, refaktoreerimine, sõltuvuste haldamine

Mitte puhas kood

Refaktoreerimine

Koodi struktuuri parandamine, nii et selle funktsionaalsus ei muutu.

Refaktoreerimine

Koodi struktuuri parandamine, nii et selle funktsionaalsus ei muutu.

Eesmärgiga, et kood oleks kergemini aru-saadavam ja muudetavam.

100% õige on vähe

- Loetavus
  - Vaatan peale ja saan aru, mida kood teeb. Ei peab mõttes käivitama
  - Ei pea korraga palju meeles pidama. (3-4 mitte 7+-2 asja)
- Muudetavus
- Struktuur, taaskasutatavus
- Testitavus

```
public void printOrdersStartingFrom(Date date) throws Exception {

  String sql = "SELECT id, number, name, quantity, price "
      + "FROM orders o "
      + "LEFT JOIN order_items oi on oi.order_id = o.id "
      + "where date > " + date;

  try ( Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {

    String previousOrderNumber = null; int counter = 1;
    while (rs.next()) {
      String number = rs.getString("number");
      if (!number.equals(previousOrderNumber)) {
        System.out.println("Tellimus number: " + number);
        previousOrderNumber = number; counter = 1;
      }

      String line = MessageFormat.format("{0}. {1} {2} {3}",
        counter++, rs.getString("name"), rs.getString("quantity"),
        rs.getString("price"));

      System.out.println(line);
    }
  }
}
```

Tellimus number 1

1. Pliiats 3 0,40
2. Paber 5 4,30

Tellimus number 2

1..

```
public void printOrdersStartingFrom(Date date) {
  List<Order> orders = dao.getOrdersFromDate(date);

  String statement = generateStatement(orders);

  System.out.println(statement);
}
```

DAO – Data Access Object

Efektiivsus?

```
public long benchmark() {
    long start = System.currentTimeMillis();

    List<Order> orders = new ArrayList<>();
    for (int i = 0; i < 10000000; i++) {
        orders.add(new Order(i));
    }

    long total = 0;
    for (Order order : orders) {
        total += order.i;
    }

    System.out.println("total: " + total);
    return (System.currentTimeMillis() - start);
}

//     100 000 - 0,04 sek
//   1 000 000 - 0,9 sek
// 10 000 000 - 18 sek
```

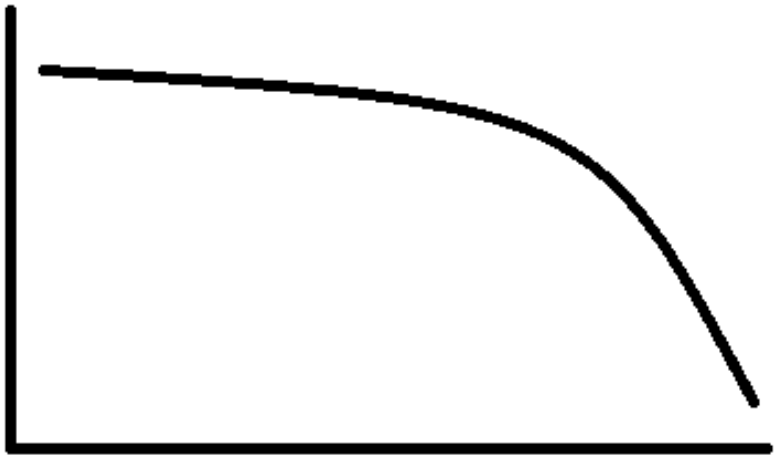
Kas tasub ära?

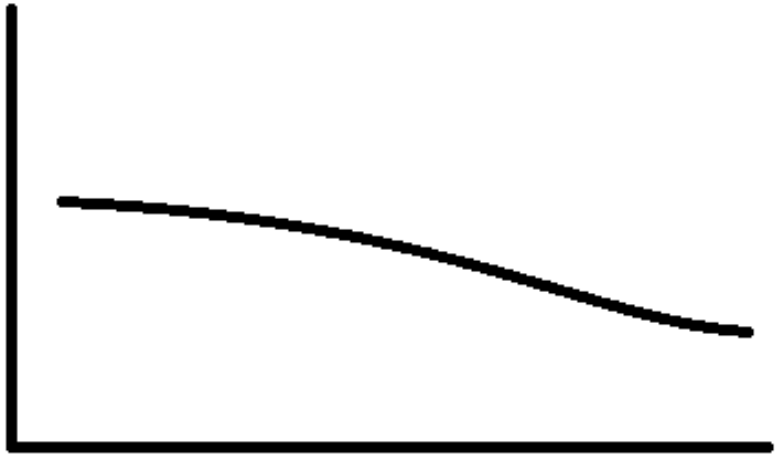
```
public void printOrdersStartingFrom(Date date) {
  List<Order> orders = dao.getOrdersFromDate(date);

  String statement = generateStatement(orders);

  System.out.println(statement);
}
```

```
public void printOrdersStartingFrom(Date date) throws Exception {

  String sql = "SELECT id, number, name, quantity, price "
      + "FROM orders o "
      + "LEFT JOIN order_items oi on oi.order_id = o.id "
      + "where date > " + date;

  try ( Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {

    String previousOrderNumber = null; int counter = 1;
    while (rs.next()) {
      String number = rs.getString("number");
      if (!number.equals(previousOrderNumber)) {
        System.out.println("Tellimus number: " + number);
        previousOrderNumber = number; counter = 1;
      }

      String line = MessageFormat.format("{0}. {1} {2} {3}",
        counter++, rs.getString("name"), rs.getString("quantity"),
        rs.getString("price"));

      System.out.println(line);
```

Vastuväited

- Mõtetu, miks töötavat asja puutuda!
- Muudetavus ja loetavus on pisut hägused asjad kellele seda vaja on?
- Refaktoreerimse eest ei saa ülemuselt/kliendilt kiita aga kui midagi ära lõhkuda on pahasti.





Vertikaal: funktsionaalsuse lisandumine
Horisontaal: aeg

Kuidas aru saada, et on vaja refaktoreerida?

- Muudatusi on raske teha
- Muudatused avaldavad laialdast mõju
- Code smells - häirivad, ebameeldivust tekitavad kohad koodis.

Long Method

```
public void printOrdersStartingFrom(Date date) throws Exception {

  String sql = "SELECT id, number, name, quantity, price "
      + "FROM orders o "
      + "LEFT JOIN order_items oi on oi.order_id = o.id "
      + "where date > " + date;

  try ( Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql)) {

    String previousOrderNumber = null; int counter = 1;
    while (rs.next()) {
      String number = rs.getString("number");
      if (!number.equals(previousOrderNumber)) {
        System.out.println("Tellimus number: " + number);
        previousOrderNumber = number; counter = 1;
      }

      String line = MessageFormat.format("{0}. {1} {2} {3}",
        counter++, rs.getString("name"), rs.getString("quantity"),
        rs.getString("price"));

      System.out.println(line);
    }
```

Long Parameter List

```
public HashMap getAccountStatement(String accountId,
      String filter,
      String startDate,
      String endDate,
      String page,
      int maxCount,
      boolean bShort)


getAccountStatement("1", filter, null, null, "1", 10, false);

getAccountStatement("1", filter, null, null, "1", 10, true);
```

Selector Arguments

```
public HashMap getAccountStatement(String accountId,
      String filter,
      String startDate,
      String endDate,
      String page,
      int maxCount,
      boolean bShort)
```

Kommentaarid

- Mõtetud kommentaarid

  i++; // suurendame i-d

  foreach (... // käime listi läbi
- Välja kommenteeritud kood
- Kommenteerida vaid juhul kui koodist jääb väheks. Mida? vs. miks?
- Isekommenteeriv kood

```
// calculate order total
double orderTotal = 0;
for (OrderItem item : order.getOrderItems()) {
        orderTotal += item.getPrice();
}
```



```
double orderTotal = getOrderTotal(order);
```

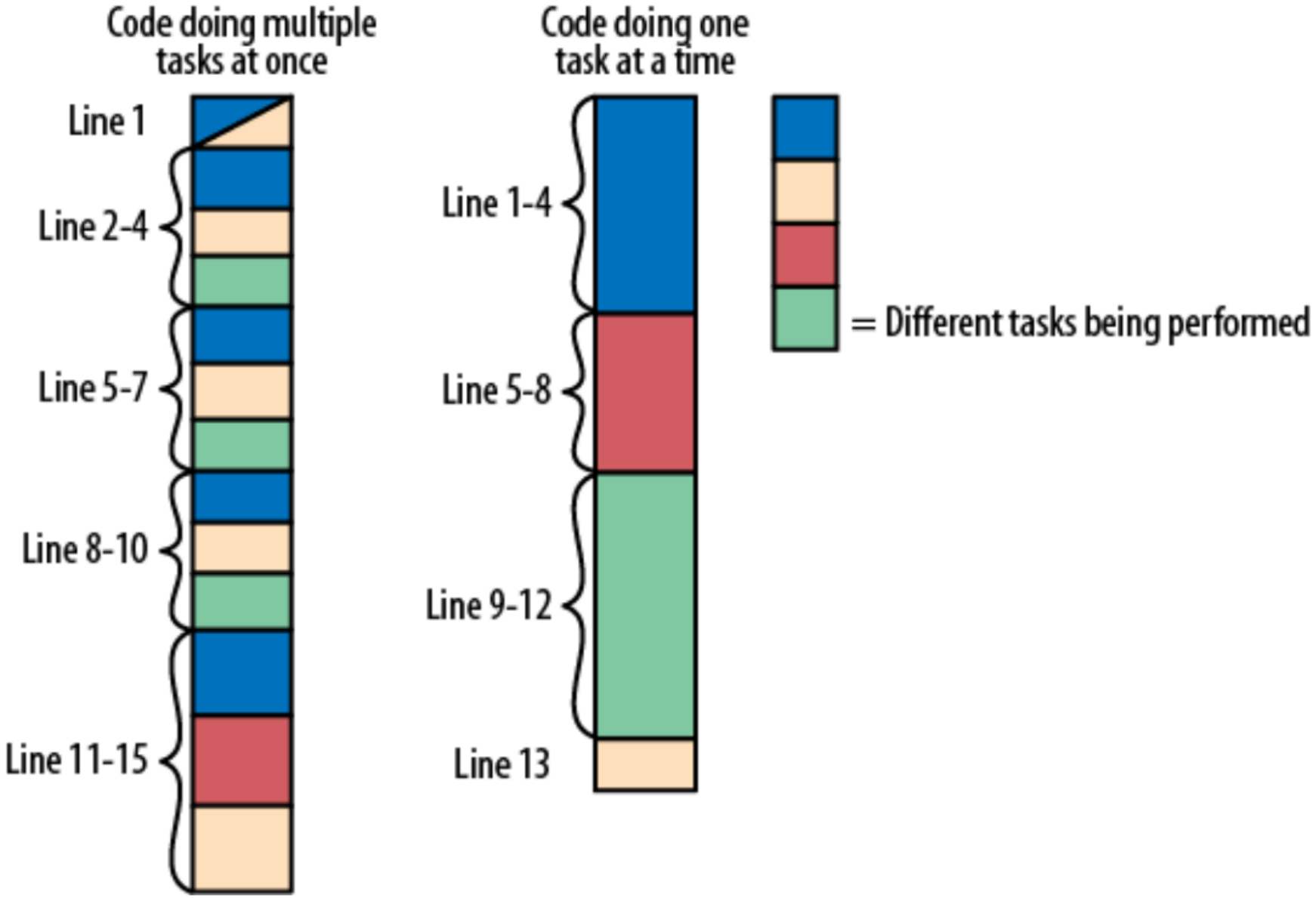
```
/*
 * Copyright (c) 1995, 2008, Oracle and/or its affiliates. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 *   - Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *
 *   - Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in the
 *     documentation and/or other materials provided with the distribution.
 *
 *   - Neither the name of Oracle or the names of its
 *     contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
 * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/**
 * The HelloWorldApp class implements an application that
 * simply prints "Hello World!" to standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
```

Pikad blokid

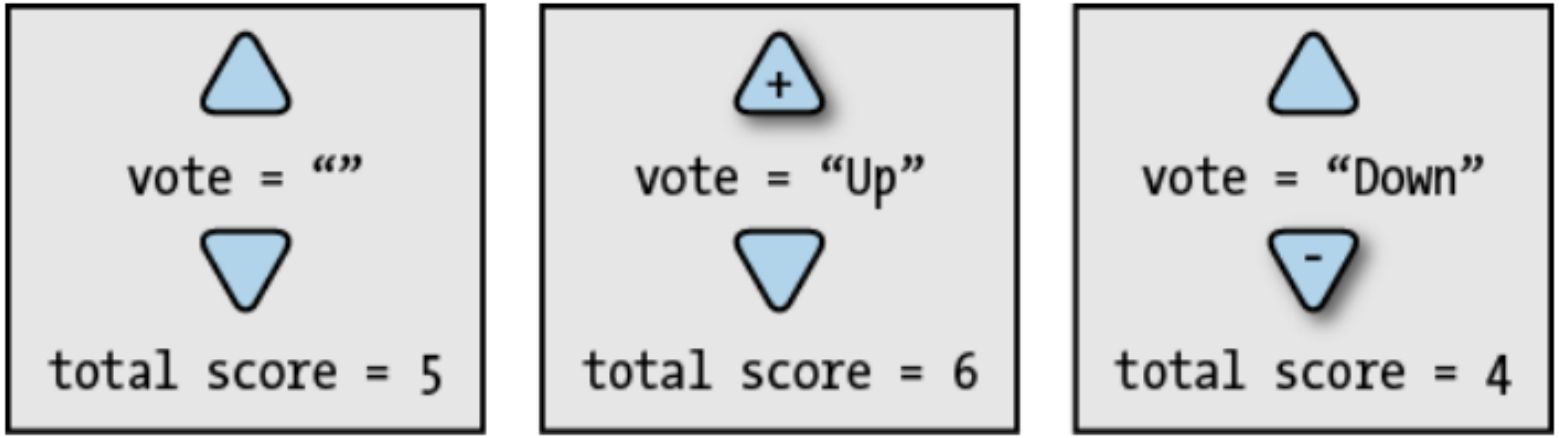
```
void transferMoney(Money money, Account from, Account to) {
  if (accessAllowed()) {

    // hulk koodi

  }
}
```

```
void transferMoney(Money money, Account from, Account to) {
  if (!accessAllowed()) return;

  // hulk koodi
}
```

Üks asi korraga



Üks asi korraga



Üks asi korraga

```
public void changeVote (String oldVote, String newVote) {

    if (newVote.equals(oldVote)) {
        if ("Up".equals(newVote)) {
            score += ("Down".equals(oldVote) ? 2 : 1);
        } else if ("Down".equals(newVote)) {
            score -= ("Up".equals(oldVote) ? 2 : 1);
        } else if ("".equals(newVote)) {
            score += ("Up".equals(oldVote) ? -1 : 1);
        }
    }
}
```

Üks asi korraga

- “Up”, “Down”, “” stringide töötlus
- Skoori muutmine

Üks asi korraga

```
public int voteValue(String vote) {
      if ("Up".equals(vote)) {
            return 1;
      } else if ("Down".equals(vote)) {
            return -1;
      } else {
            return 0;
      }
}
```

```
public void changeVote(String oldVote, String newVote) {
     score -= voteValue(oldVote);  // remove the old vote
     score += voteValue(newVote);  // add the new vote
}
```

Refactorings

- meetmed ebakõlade eemaldamiseks

Rename method

- method1(), z()
- processInvoice(Invoice i)
- Principle of Least Surprise
  - get/set
  - getItems(List list)

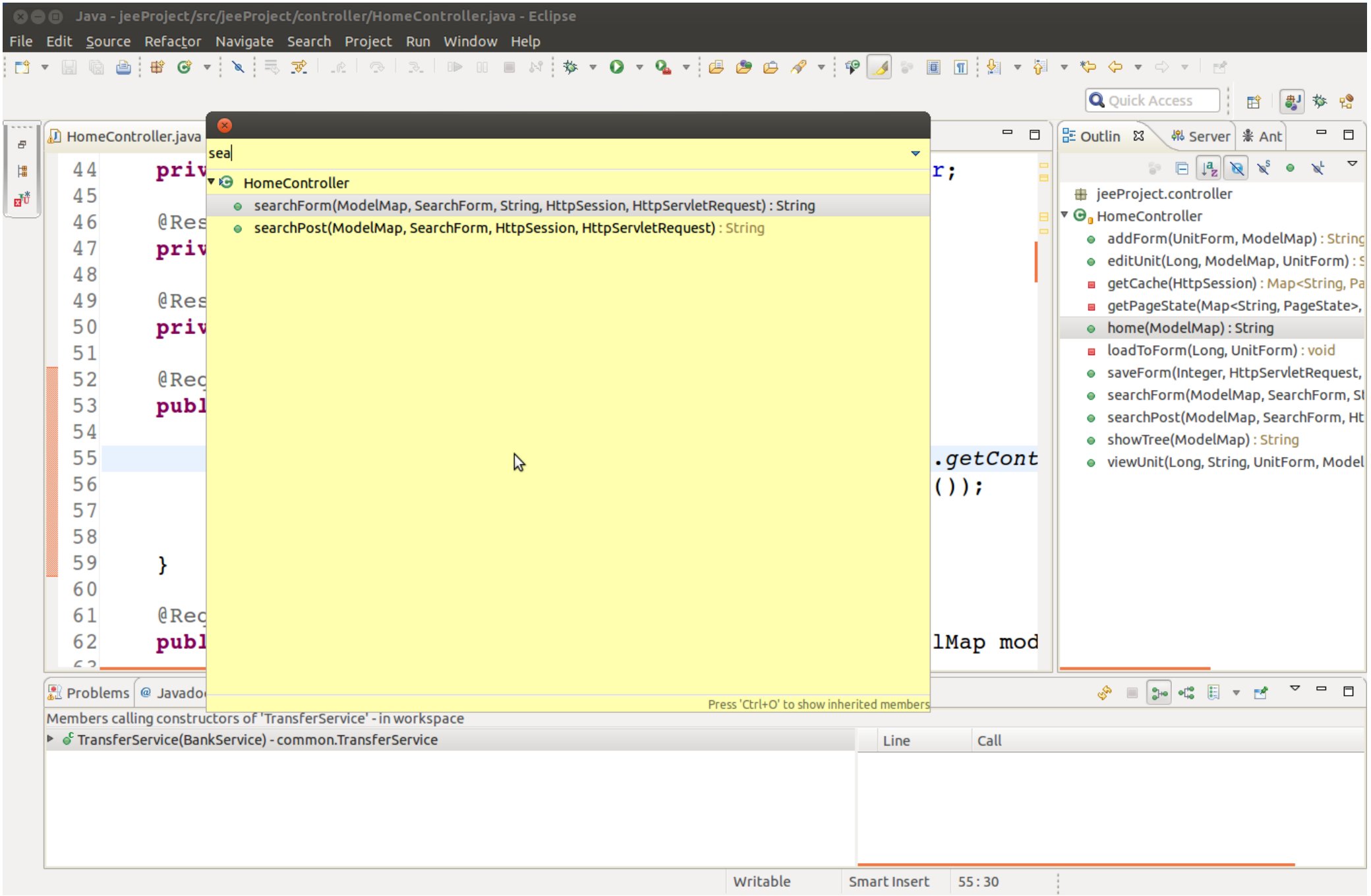
  vs.

  collectItemsTo(List list)

Extract method

```
void printOwing(double amount) {
  printBanner();

  // print details
  System.out.println("name:" + name);
  System.out.println("amount" + amount);
}
```

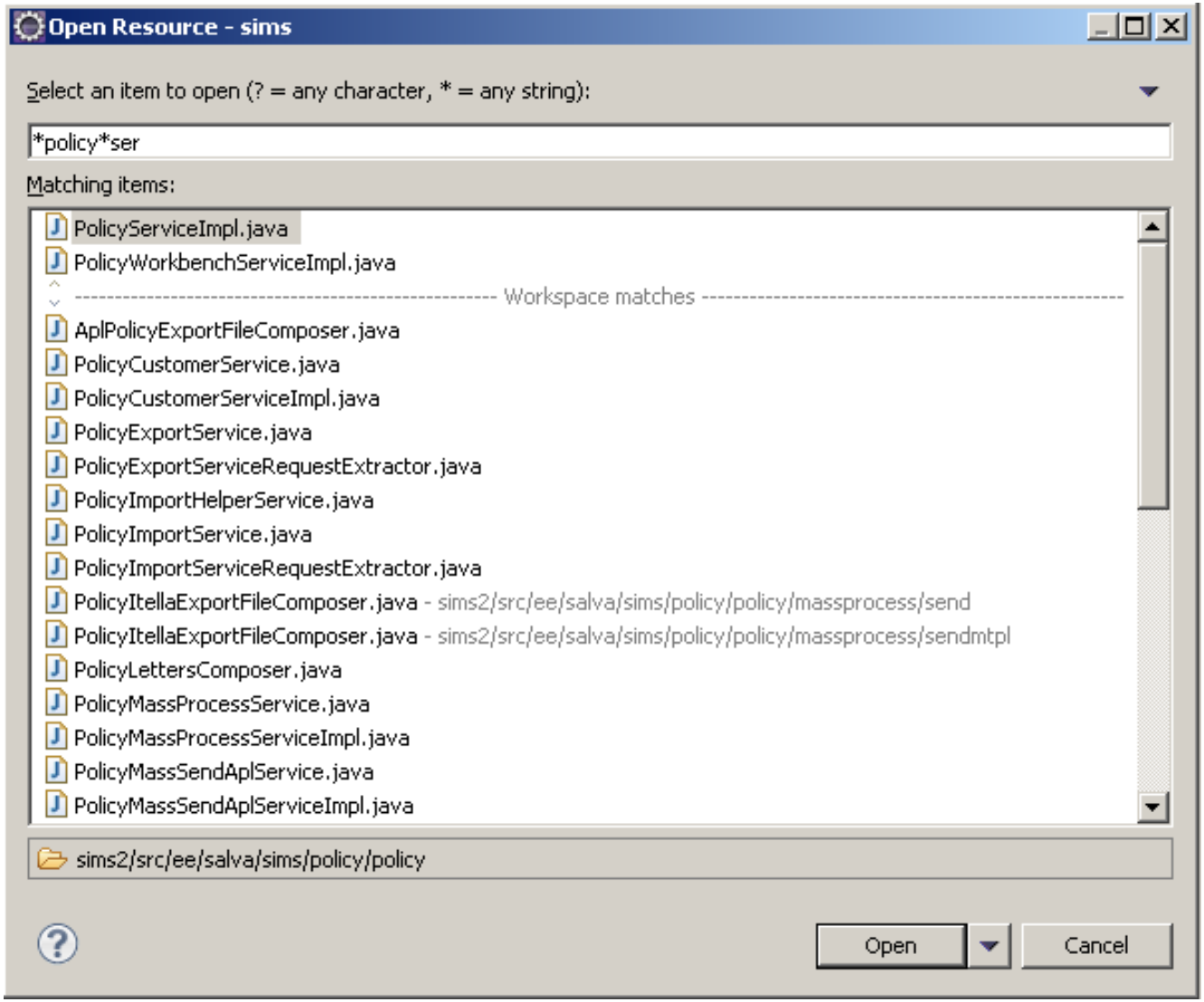
```
void printOwing(double amount) {
  printBanner();
  printDetails(amount);
}
```


Java - jeeProject/src/jeeProject/controller/HomeController.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
HomeController.java
sea
HomeController
searchForm(ModelMap, SearchForm, String, HttpSession, HttpServletRequest) : String
searchPost(ModelMap, SearchForm, HttpSession, HttpServletRequest) : String
Press 'Ctrl+O' to show inherited members
Outlin
Server
Ant
jeeProject.controller
HomeController
addForm(UnitForm, ModelMap) : String
editUnit(Long, ModelMap, UnitForm) :
getCache(HttpSession) : Map<String, Pa
getPageState(Map<String, PageState>,
home(ModelMap) : String
loadToForm(Long, UnitForm) : void
saveForm(Integer, HttpServletRequest,
searchForm(ModelMap, SearchForm, St
searchPost(ModelMap, SearchForm, Ht
showTree(ModelMap) : String
viewUnit(Long, String, UnitForm, Model
Problems
Javado
Members calling constructors of 'TransferService' - in workspace
TransferService(BankService) - common.TransferService
Line
Call
Writable
Smart Insert
55 : 30

Extract Class

- Kui klass teeb mitut erinevat asja
- Loetavus,
- Nimetamine

Klasside nimetamine



Introduce Parameter Object

```
amountReceivedIn(Date start, Date end);

amountReceivedIn(Range between);
```

Decompose Conditional

```
private double getPrice() {
  if (date.after(SUMMER_START)
              && date.before(SUMMER_END)
              && roomType == NORMAL) {
    ...
  }
}
```

```
private double getPrice() {
  if (isSummer() && roomType == NORMAL) {
    ...
  }
}
```

```
if ((status == null || status.equals(x))
            && ((type == y || type == z)
            || size > 20)) {
      ...
}
```

http://refactoring.com/catalog/index.html

Refactorings in Alphabetical Order

This is a simple list of refactorings both from the original book and some later sources. Sad
UPDATED with have some extra material to what's in the Refactoring book. Refactorings mar
references to refactorings described elsewhere.

| Russian | German |

- Add Parameter
- Change Bidirectional Association to Unidirectional
- Change Reference to Value
- Change Unidirectional Association to Bidirectional UPDATED
- Change Value to Reference
- Collapse Hierarchy
- Consolidate Conditional Expression
- Consolidate Duplicate Conditional Fragments UPDATED
- Convert Dynamic to Static Construction *by Gerard M. Davison* NEW
- Convert Static to Dynamic Construction *by Gerard M. Davison* NEW
- Decompose Conditional
- Duplicate Observed Data
- Eliminate Inter-Entity Bean Communication *(Link Only)*
- Encapsulate Collection
- Encapsulate Downcast
- Encapsulate Field

Ülesanne 4: refaktoreerimine

Protsess

- Äkki kohe “õigesti” kirjutada?
- Väikesed sammud

Mitte puhas kood

- Väljaprint ja värvipliiatsid
- Proovi refaktoreerimine

Sõltuvuste haldamine

Stub

Stub - tupikprogramm, jupats

Puuduva või probleemse koodi asendamine.

Mittesoovitavad kõrvalefektid

```
try {

    Message message = new MimeMessage(session);
    message.setFrom(new InternetAddress("from-email@gmail.com"));
    message.setRecipients(Message.RecipientType.TO,
        InternetAddress.parse("to-email@gmail.com"));
    message.setSubject("Testing Subject");
    message.setText("Testing content");

    Transport.send(message);

} catch (MessagingException e) {
    throw new RuntimeException(e);
}
```

Mingi osa ei toimi

```
public Money getOrderTotal(String orderId) {
  List<Money> orderAmounts = dao.getOrderAmounts(orderId);

  Money total = new Money(0, "EUR");

  for (Money amount : orderAmounts) {
    total = total.plus(amount);
  }

  return total;
}
```

Testide aeglus

```
public Money getOrderTotal(String orderId) {
  List<Money> orderAmounts = dao.getOrderAmounts(orderId);

  Money total = new Money(0, "EUR");

  for (Money amount : orderAmounts) {
    total = total.plus(amount);
  }

  return total;
}
```

Raskesti korratavad olekud

```
try {
    socket = new Socket("comp1", 4321);
    out = new PrintWriter(socket.getOutputStream(), true);
    in = new InputStreamReader(socket.getInputStream());
} catch (UnknownHostException e) {
    // ...
} catch (IOException e) {
    // ...
}
```

Mitte ühene käitumine

```
List<Integer> list = Arrays.asList(1, 2, 3);

Collections.shuffle(list, new Random());
```

Stub-ide kasutamise eeldused

- Päris igat koodijuppi ära vahetada ei saa

```
public Money getOrderTotal(String orderId) {

  // ... Connection con =
  //           DriverManager.getConnection( …
  // ...

  Money total = new Money(0, "EUR");

  for (Money amount : orderAmounts) {
    total = total.plus(amount);
  }

  return total;
}
```

Stub-ide kasutamise eeldused

- Kood peab olema eraldi objektis või vähemalt eraldi meetodis.

Sõltuvus objektist

```
public Money getOrderTotal(String orderId) {
  Dao dao = new DbDao();

  List<Money> orderAmounts = dao.getOrderAmounts(orderId);

  Money total = new Money(0, "EUR");

  for (Money amount : orderAmounts) {
    total = total.plus(amount);
  }

  return total;
}
```

Sõltuvus objektist

```
private Dao dao;

public Money getOrderTotal(String orderId) {

  List<Money> orderAmounts = dao.getOrderAmounts(orderId);

  Money total = new Money(0, "EUR");

  for (Money amount : orderAmounts) {
    total = total.plus(amount);
  }

  return total;
}
```

Stub implementatsioonina

```
interface Dao {
  List<Money> getOrderAmounts(String orderId);
}
```

```
class TestDao implements Dao {
  private List<Money> getOrderAmounts(String orderId) {

    return Arrays.asList(new Money(3, "EUR"),
                         new Money(1, "EEK"));
  }
}
```

Stub implementatsioonina puudus

```
interface Dao {
  List<Money> getOrderAmounts(String orderId);
  List<Order> getOrders();
  Money getOrderTotal(Order order);
  // etc..
}

class TestDao implements Dao {
  private List<Money> getOrderAmounts(String orderId) {
    return Arrays.asList(new Money(3, "EUR"),
                         new Money(1, "EEK"));
  }

  private List<Order> getOrders() {
      return null;
  }
  //etc..
}
```

Stub alamklassina

```
public class DbDao {
  public List<Order> getOrders() {
    // connect to db, etc...
    return a real list of orders;
  }
}

class TestDao extends DbDao {
  @Override
  public List<Order> getOrders() {
    return Arrays.asList(
       new Order("Pen", 1), new Order("Paper", 2));
  }
}
```

Stub anonüümse alamklassina

```
@Test
public void anonymusSubclassTest() throws Exception {
  final Order order1 = new Order("Pen", 1);
  final Order order2 = new Order("Paper", 2);

  DbDao dao = new DbDao() {
    @Override
    public List<Order> getOrders() {
      return Arrays.asList(order1, order2);
    }
  };

  OrderService service = new OrderService()
  service.setDao(dao);

  Order lastOrder = service.getLastOrder();
  ...
}
```

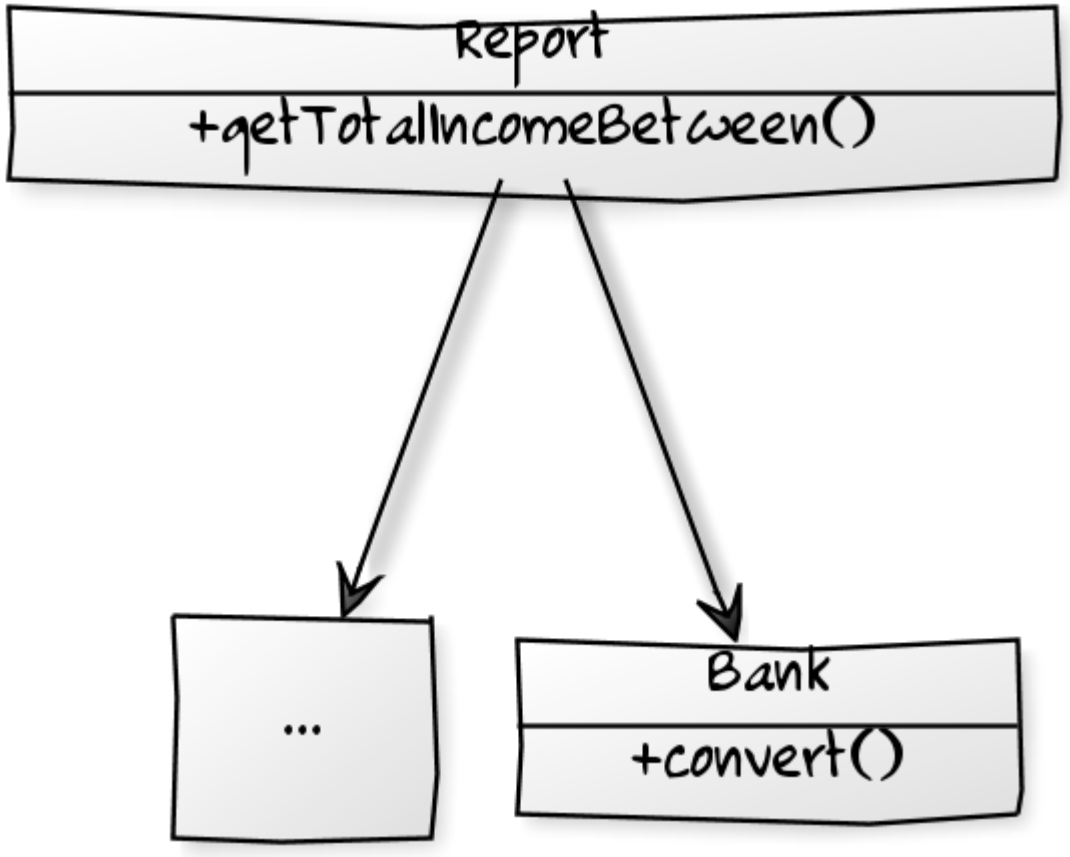
Sõltuvus meetodist

```
public class EmployeePayCalculator {

	public BigDecimal calculatePay(Long employeeId) {
		Employee e = fetchForId(employeeId);

		// do calculations

		return ...
	}

	// private
	protected Employee fetchForId(Long employeeId) {
		// get it from database and return
	}
}
```

Sõltuvus meetodist

```
final Employee employee = new Employee();

// init employee

EmployeePayCalculator calc = new EmployeePayCalculator() {

	@Override
	protected Employee fetchForId(Long employeeId) {
		return employee;
	}
};

calc.calculatePay(...
```

Ülesanne 5: stubide kasutamine



```java
@Test
public void testReportTotalIncome() {
    Report report = new Report(new Bank());

    Money total = report.getTotalIncomeBetween(null, null);

    assertThat(total, is(new Money(2, "EUR")));
}
```

public class Report ...

```
public Money getTotalIncomeBetween(
        Date startDate, Date endDate) {

    List<Money> incomes = getIncomesBetween(
                                    startDate, endDate);

    Money total = new Money(0, "EUR");

    for (Money each : incomes) {
        total = total.plus(bank.convert(each, "EUR"));
    }

    return total;
}

protected List<Money> getIncomesBetween(
        Date startDate, Date endDate) {

    throw new IllegalStateException("not implemented");
}
```

```
class Bank {

    public Money convert(Money money, String toCurrency) {
        throw new IllegalStateException("not implemented");
    }

}
```

Mock

- Käitumise kontrollimiseks. So. kuidas testitav komponent oma keskkonnaga suhtleb.
- Samas täidab ka stub-i osa, tagastades fikseeritud väärtusi

TransferService

Käitumise kontrollimine

```
public void transferMoney(int amount,
    String formAccount, String toAccount) {

  if (amount <= 0 || formAccount.equals(toAccount)) return;

  if (bankService.getBalance(formAccount) >= amount) {
    bankService.transfer(amount, formAccount, toAccount);
  }
}
```

Käitumise kontrollimine

```
class TransferService {

  private BankService bankService;

  public TransferService(BankService bankService) {
    this.bankService = bankService;
  }

  public void transferMoney(int amount,
      String formAccount, String toAccount) {

    if (amount <= 0 || formAccount.equals(toAccount)) return;

    if (bankService.getBalance(formAccount) >= amount) {
      bankService.transfer(amount, formAccount, toAccount);
    }
  }
}
```

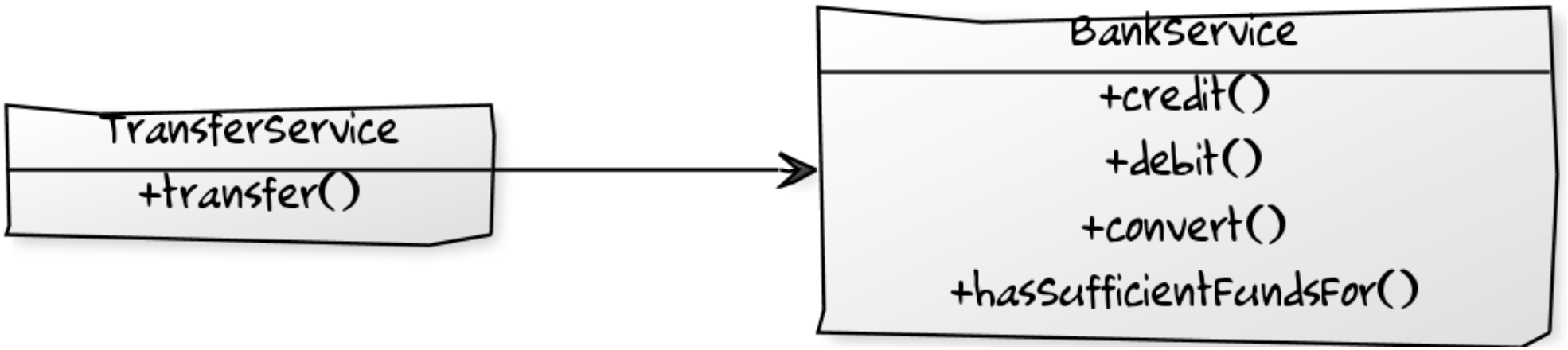
Käitumise kontrollimine


```
@Test
public void transferInvokesBalanceCheck() {
      MockBankService mockBankService = new MockBankService();

      TransferService transferService =
            new TransferService(mockBankService);

      transferService.transferMoney(10, "123", "456");

      assertTrue(mockBankService.wasGetBalanceCalled());
}
```

Mock-i näide

```
public class MockBankService implements BankService {

	private boolean getBalanceCalled = false;

	@Override
	public int getBalance(String formAccount) {
		getBalanceCalled = true;
		return 100;
	}

	public boolean wasGetBalanceCalled() {
		return getBalanceCalled;
	}

	@Override
	public void transfer(int amount, String formAccount,
				String toAccount) {
	}
}
```

Ülesanne 6: isetehtud mock-id



```
@Test
public void transferWithCurrencyConversion() {

    TestableBankService bankService = new TestableBankService();

    TransferService transferService =
                                new TransferService(bankService);

    transferService.transfer(new Money(15, "EEK"), "123", "456");

    assertTrue(bankService.wasCreditCalledWith(
            new Money(15, "EEK"), "123"));

    assertFalse(bankService.wasCreditCalledWith(
            new Money(0, "EEK"), ""));

    ...

}
```

TransferService.transfer()

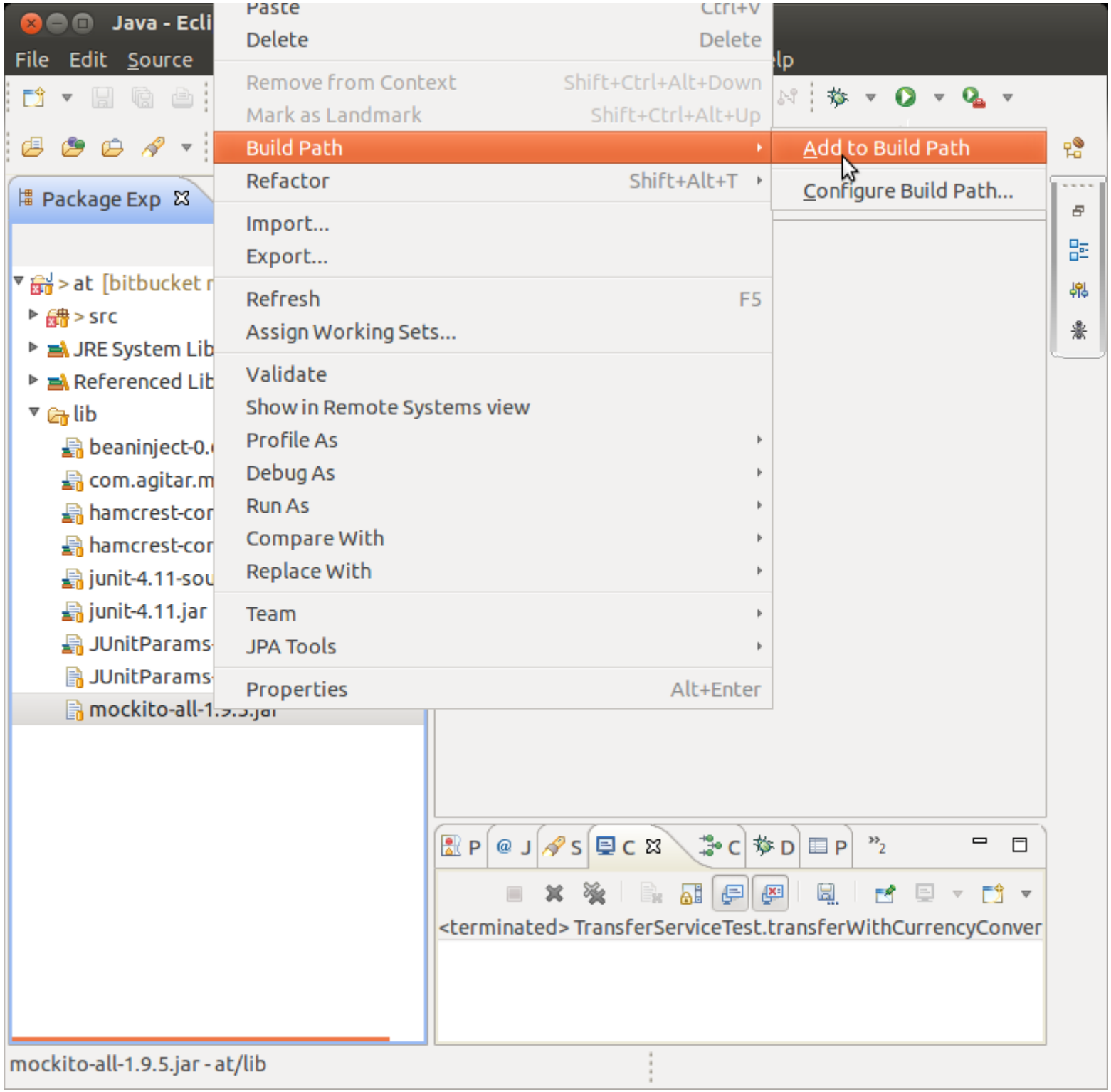
```
public void transfer(Money money, String fromAccount, String toAccount) {

    // konverteeri summa krediteeritava konto valuutassse
    // ja krediteeri

    String fromCurrency = bankService.getAccountCurrency(fromAccount);
    Money creditAmountConverted = bankService.convert(money, fromCurrency);

    if (!bankService.hasSufficientFundsFor(creditAmountConverted))
        return;

    bankService.credit(creditAmountConverted, fromAccount);

    // konverteeri summa debiteeritava konto valuutassse
    // ja debiteeri

    String toCurrency = bankService.getAccountCurrency(toAccount);
    Money debitAmountConverted = bankService.convert(money, toCurrency);
    bankService.debit(debitAmountConverted, toAccount);
}
```

```
@Override
public void credit(Money money, String fromAccount) {
    throw new IllegalStateException("not implemented");
}

public boolean wasCreditCalledWith(
    Money money, String account) {

    throw new IllegalStateException("not implemented");
}
```

Mock-imise raamistikud

- Palju erinevaid valikuid erineva võimekusega
- Kasutame Mockito-t

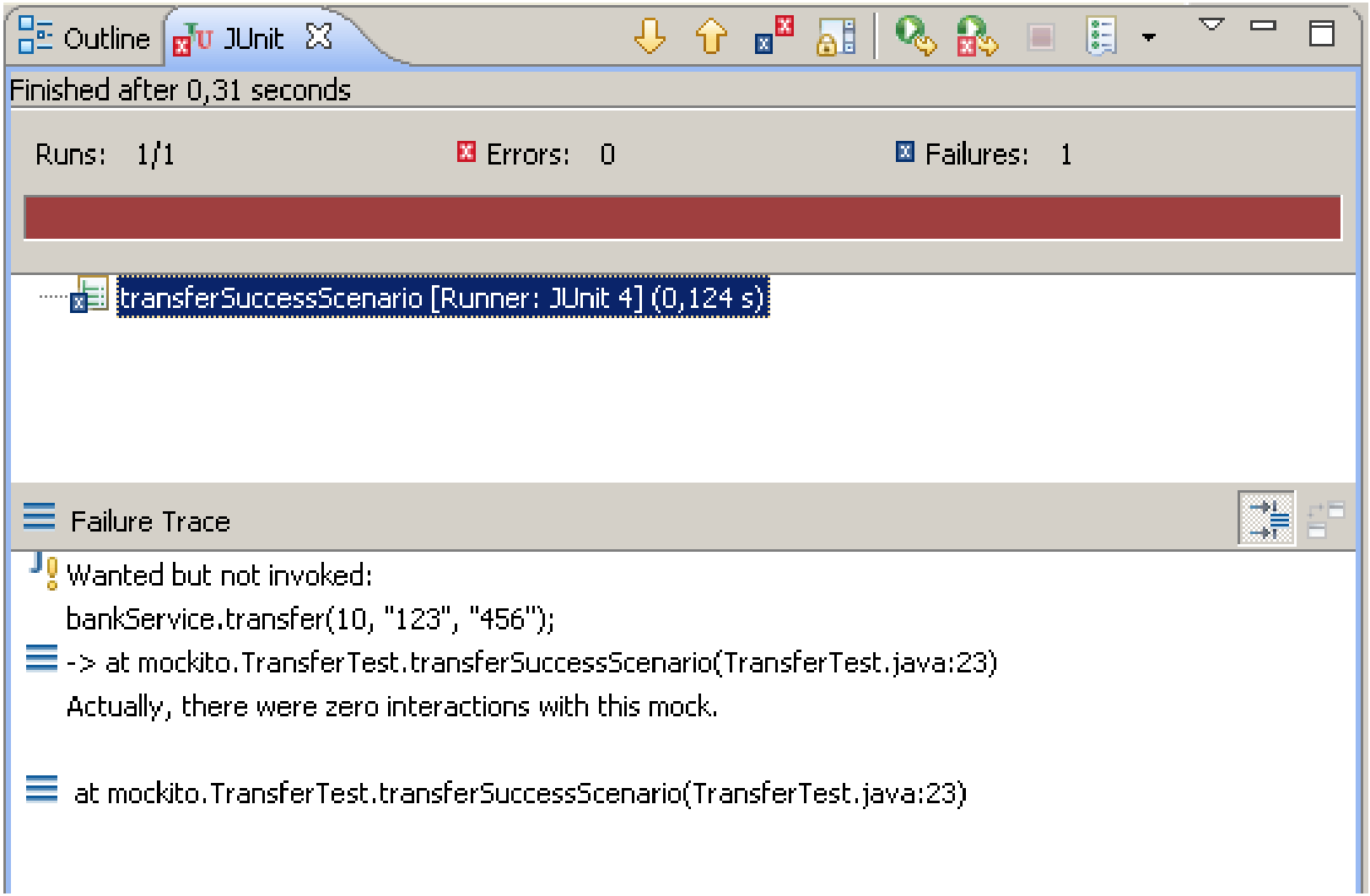
  http://code.google.com/p/mockito/
- Mockito võimaldab kasutada lisana PowerMock-i


Java - Ecli
File Edit Source
Paste Ctrl+V
Delete Delete
Remove from Context Shift+Ctrl+Alt+Down
Mark as Landmark Shift+Ctrl+Alt+Up
Build Path
Refactor Shift+Alt+T
Import...
Export...
Refresh F5
Assign Working Sets...
Validate
Show in Remote Systems view
Profile As
Debug As
Run As
Compare With
Replace With
Team
JPA Tools
Properties Alt+Enter
Add to Build Path
Configure Build Path...
Package Exp
> at [bitbucket
> src
JRE System Lib
Referenced Lib
lib
beaninject-0.
com.agitar.m
hamcrest-cor
hamcrest-cor
junit-4.11-sou
junit-4.11.jar
JUnitParams
JUnitParams
mockito-all-1.9.5.jar
<terminated> TransferServiceTest.transferWithCurrencyConver
mockito-all-1.9.5.jar - at/lib

```
interface BankService {

   int getBalance(String formAccount);

   void transfer(int amount,
        String formAccount, String toAccount);
}
```

```
import static org.mockito.Mockito.*;

...

private BankService mockBankService = mock(BankService.class);

@Test
public void transferSuccessScenario() {
  TransferService transferService = new TransferService(mockBankService);

  when(mockBankService.getBalance("123")).thenReturn(1000);

  transferService.transferMoney(10, "123", "456");

  verify(mockBankService).transfer(10, "123", "456");
}
```

```
public void transferMoney(int amount,
    String formAccount, String toAccount) {

  if (amount <= 0 || formAccount.equals(toAccount)) return;

  if (bankService.getBalance(formAccount) >= amount) {
    bankService.transfer(amount, formAccount, toAccount);
  }
}
```


Outline
JUnit
Finished after 0,31 seconds
Runs: 1/1
Errors: 0
Failures: 1
transferSuccessScenario [Runner: JUnit 4] (0,124 s)
Failure Trace
Wanted but not invoked:
bankService.transfer(10, "123", "456");
-> at mockito.TransferTest.transferSuccessScenario(TransferTest.java:23)
Actually, there were zero interactions with this mock.
at mockito.TransferTest.transferSuccessScenario(TransferTest.java:23)

```
@Test
public void transferingNegativeAmountFails() {

  TransferService transferService = new TransferService(mockBankService);

  transferService.transferMoney(-1, "123", "456");

  verify(mockBankService, never()).getBalance(anyString());
}
```

```
public void transferMoney(int amount,
    String formAccount, String toAccount) {

  if (amount <= 0 || formAccount.equals(toAccount)) return;

  if (bankService.getBalance(formAccount) >= amount) {
    bankService.transfer(amount, formAccount, toAccount);
  }
}
```

```
@Test
public void transferFailsWhenNotEnoughFunds() {
  TransferService transferService = new TransferService(mockBankService);

  when(mockBankService.getBalance("123")).thenReturn(5);

  transferService.transferMoney(10, "123", "456");

  verify(mockBankService, never()).transfer(
              anyInt(), anyString(), anyString());
}
```

```
  public void transferMoney(int amount,
      String formAccount, String toAccount) {

    if (amount <= 0 || formAccount.equals(toAccount)) return;

    if (bankService.getBalance(formAccount) >= amount) {
      bankService.transfer(amount, formAccount, toAccount);
    }
  }
```

Väljakutsete kontrollimine

```
List mockedList = mock(List.class);

mockedList.add("one");
mockedList.clear();

verify(mockedList).add("one");
verify(mockedList).clear();
```

Stub-id

```
List mockedList = mock(List.class);

when(mockedList.get(0)).thenReturn("first");
when(mockedList.get(1)).thenThrow(new RuntimeException());

System.out.println(mockedList.get(0));
System.out.println(mockedList.get(1));
System.out.println(mockedList.get(-1));
```

Määramata argumendid

```
when(mockedList.get(anyInt())).thenReturn("element");

System.out.println(mockedList.get(1));
```

Ülesanne 7: Mockito kasutamine

Klassikaline vs mockist lähenemine