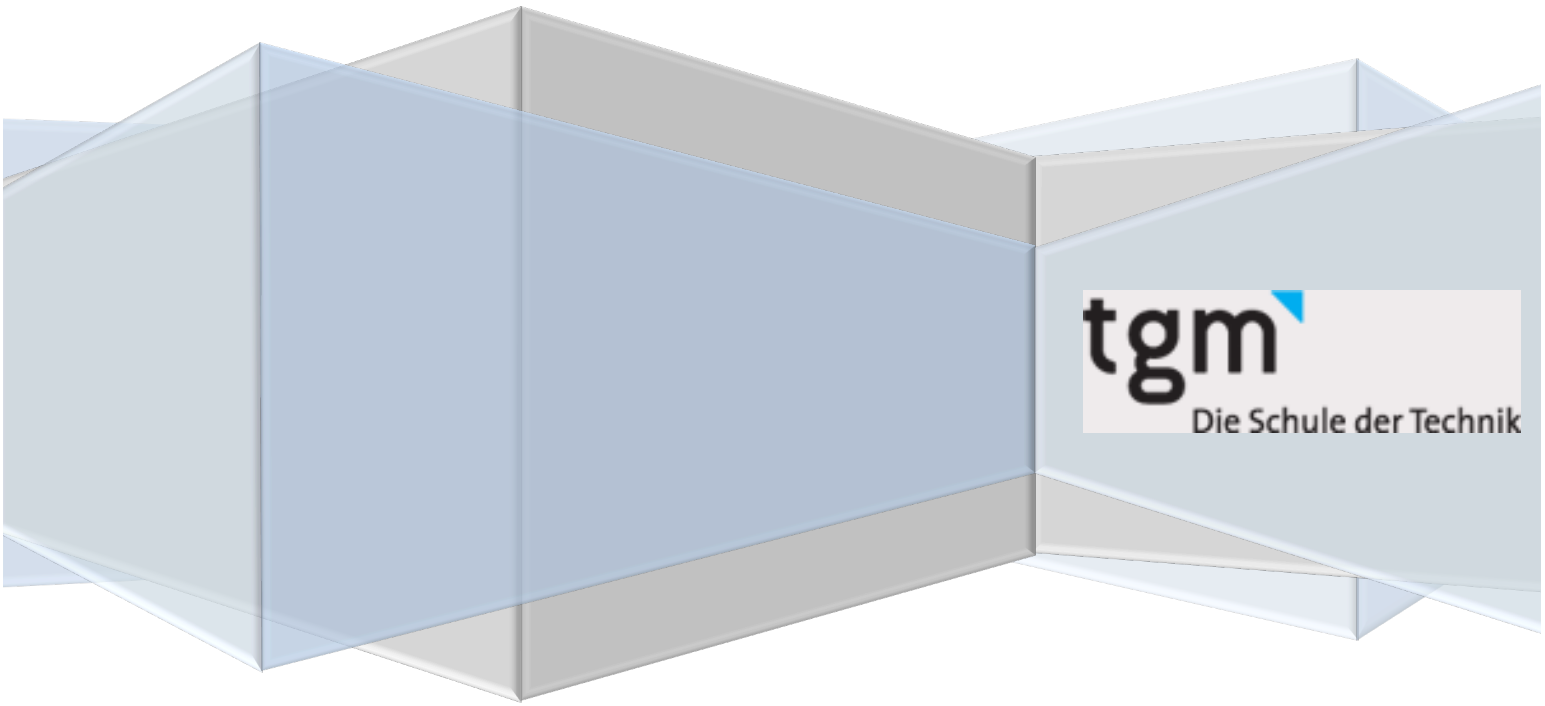
SEW

Nebenläufige Roboterfabrik

Fock, Hackenberger, Tiryaki

Schuljahr 2014/15



Inhalt

1. Aufgabenstellung / Requirementsanalyse

Es soll eine Spielzeugroboter-Fabrik simuliert werden. Die einzelnen Bestandteile des Spielzeugroboters (kurz Threadee) werden in einem Lager gesammelt. Dieses Lager wird als Verzeichnis und die einzelnen Elementtypen werden als Files im Betriebssystem abgebildet. Der Lagermitarbeiter verwaltet regelmäßig den Ein- und Ausgang des Lagers um Anfragen von Montagemitarbeiter und Kunden zu beantworten. Die Anlieferung der Teile erfolgt durch Ändern von Files im Verzeichnis, eine Lagerung fertiger Roboter ebenso.

Ein Spielzeugroboter besteht aus zwei Augen, einem Rumpf, einem Kettenantrieb und zwei Armen.

Die Lieferanten schreiben ihre Teile ins Lager-File mit zufällig (PRNG?) erstellten Zahlenfeldern. Die Art der gelieferten Teile soll nach einer bestimmten Zeit gewechselt werden.

Die Montagemitarbeiter müssen nun für einen "Threadee" alle entsprechenden Teile anfordern und diese zusammenbauen. Der Vorgang des Zusammenbauens wird durch das Sortieren der einzelnen Ganzzahlenfelder simuliert. Der fertige "Threadee" wird nun mit der Mitarbeiter-ID des Monteurs versehen.

Es ist zu bedenken, dass ein Roboter immer alle Teile benötigt um hergestellt werden zu können. Sollte ein Monteur nicht alle Teile bekommen, muss er die angeforderten Teile wieder zurückgeben um andere Monteure nicht zu blockieren. Fertige "Threadee"s werden zur Auslieferung in das Lager zurück gestellt.

Alle Aktivitäten der Mitarbeiter muss in einem Logfile protokolliert werden. Verwenden Sie dazu Log4J [1].

Die IDs der Mitarbeiter werden in der Fabrik durch das Sekretariat verwaltet. Es dürfen nur eindeutige IDs vergeben werden. Das Sekretariat vergibt auch die eindeutigen Kennungen für die erstellten "Threadee"s.

Tipps und Tricks

Verwenden Sie (optional) für die einzelnen Arbeiter das ExecutorService mit ThreadPools. Achten Sie, dass die Monteure nicht "verhungern". Angeforderte Ressourcen müssen auch sauber wieder freigegeben werden.

Beispiel für Teile-Files

-- "auge.csv"
Auge,11,24,3,4,25,6,8,8,9,10,11,12,13,14,15,16,17,18,195,5
Auge,91,62,3,4,54,6,7,8,9,10,11,12,13,14,15,16,17,18,119,32
Auge,91,62,3,4,54,6,7,8,9,10,11,12,13,14,15,16,17,18,119,520

-- "rumpf.csv"
Rumpf,91,62,3,4,54,6,7,8,9,10,11,12,13,14,15,16,17,18,119,21

Beispiel für Threadee-File

-- "auslieferung.csv"
Threadee-ID123,Mitarbeiter-
ID231,Auge,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Auge,1,2,3,4,5,6
,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Rumpf,1,2,3,4,5,6,7,8,9,10,11,12,13,14
,15,16,17,18,19,20,Kettenantrieb,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,
19,20,Arm,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Arm,1,2,3,4,5,6,7
,8,9,10,11,12,13,14,15,16,17,18,19,20
Threadee-ID124,Mitarbeiter-
ID231,Auge,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Auge,1,2,3,4,5,6
,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Rumpf,1,2,3,4,5,6,7,8,9,10,11,12,13,14
,15,16,17,18,19,20,Kettenantrieb,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,
19,20,Arm,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Arm,1,2,3,4,5,6,7
,8,9,10,11,12,13,14,15,16,17,18,19,20

2. Zeitabschätzung / Zeitaufzeichnung

Planung

Aufgabe	Abgeschätzt	Aktuell
UML-Diagramm	1.5h	2h
UML-Diagramm Verbessert		1h

Organisation

Aufgabe	Abgeschätzt	Aktuell
Git	0.2h	0.2h

Implementierung

Aufgabe	Abgeschätzt	Aktuell
Klassen & Methoden	8h	9h
CLI	0.6h	0.6h
Log4J	1 h	1h

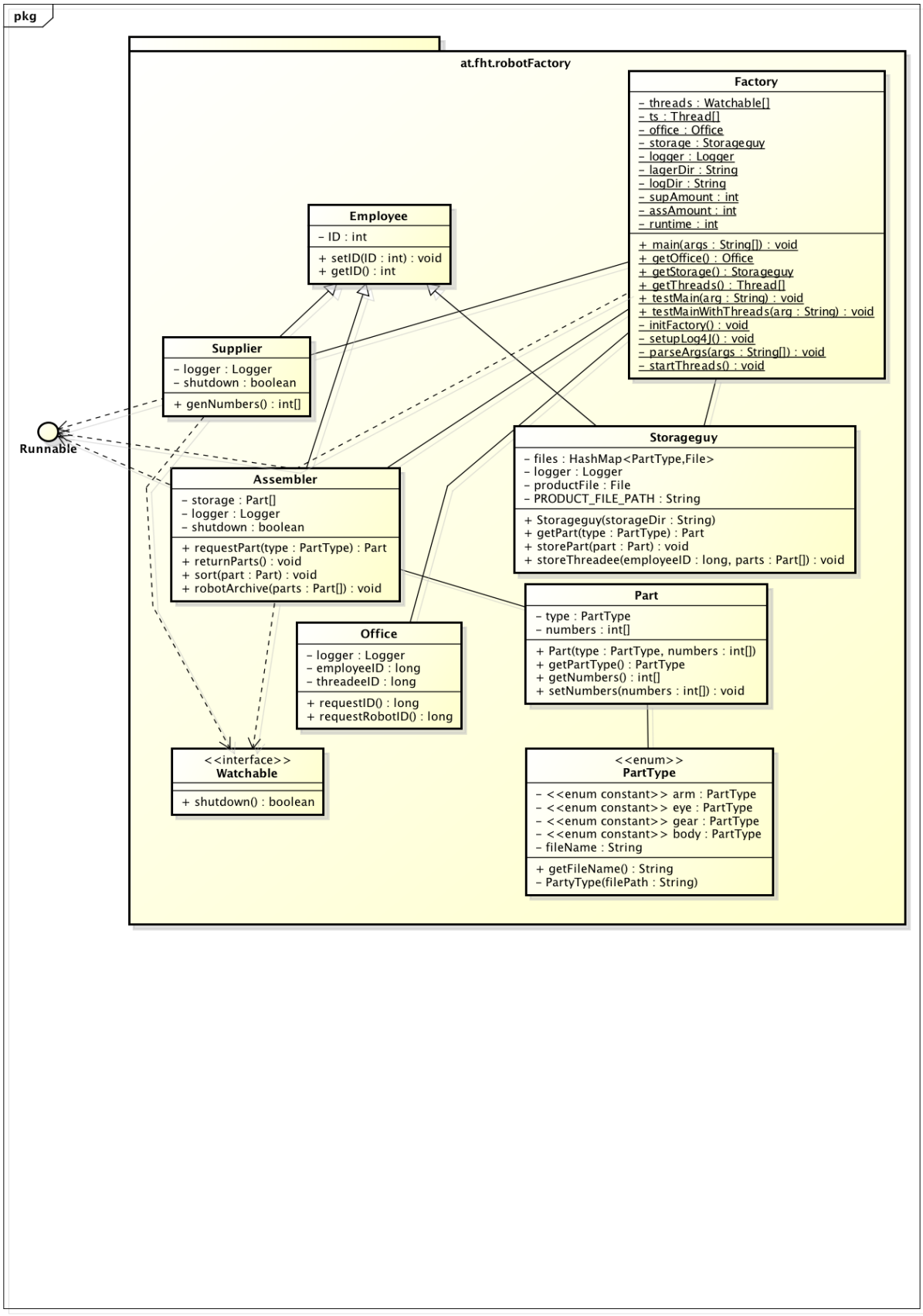
Dokumentation

Aufgabe	Abgeschätzt	Aktuell
Dokumentation des Projektes	2h	3h

Gesamt

Abgeschätzt : Aktuell	**13.3h**	**16.8**

3. Klassendiagramm



4. Things we've done

Log4j

Verwendung von Log4J für Logging

Commons CLI

API zur Verarbetiung von Command Line options

JUnit

Unit Testing

OpenCSV

API zum parsen von CSV Dateien.

Klassen + Methoden

Office

- Vergibt die IDs der Mitarbeiter
- Vergibt die IDs der Threadees

Employee

- Wird von allen Arbeitern (Storageguy, Assembler und Supplier) geerbt.

Assembler (extends Employee implements Runnable, Watchable)

- Fordert einen Part mit einem expliziten PartType + generierten Nummern, vom Storageguy an.
- Gibt alle Parts zurück die er angefragt hatte, falls eines bei seiner „Bestellung“ fehlt.
- Sortiert die Nummern, die bei einem Part dabei sind, als ein symbolisches Zeichen für das Zusammensetzen eines Threadees.
- Lagert den zusammengesetzten Threadee via Storageguy in ein .csv File ein.
- Run()
    - Fordert alle Parts für einen Threadee an
    - Sortiert jeden einzelnen Part
    - Schickt den „assemblierten“ Threadee an den Storageguy damit dieser ihn einlagert

Storageguy (extends Employee)

- Holt sich einen Part, mit einem expliziten PartType aus einem dem entsprechenden .csv flie.

- Fügt einen Part, mit einem explizitem PartType, einem dem entsprechenden .csv File hinzu.
- Fügt einen Threadee einem .csv hinzu, welches nur für die Lagerung der Threadees dient.

Supplier (extends Employee implements Runnable, Watchable)

- Generiert Nummern für einen Part
- Run()
    - Liefert dem Storageguy, nach einem Zufallsprinzip, den PartTyp & selbst generierte Nummern.

Factory

- Mit Commons CLI werden Command Line Options geparst
- Macht das Setup für Log4J
- Initialisiert den Storageguy & das Office.
- Initialisiert und started je nach Parameter die Arbeiter-Threads(Supplier und Assembler)
- Ist der Watchdog für alle Threads im Programm
- Run()
    - Stoppt nach einer über die Command Line Options definierte Zeit alle Threads

Part

- Ein Part besteht aus einem PartTyp (ARM, EYE,…) und aus 20 zufällig generierten Zahlen.
        - Getter- Setter- Methoden

PartType

- Ist ein Enum der verschiedenen PartTypes

5. Lessons learned

Threads

Threads können parallel arbeiten ➔ gleichzeitig einen Programmcode ausführen.

Bezeichnungen für Threads

- Aktivitätsträger
- Leichtgewichtiger Prozess

Es gibt 2 Arten der Implementierung eines Threads

- Implementation der Klasse Thread ➔ (implements Runnable)
  - Diese Methode ist "besser" da man nicht die Vererbungshirarchie blockiert
- Erweiterung der Klasse Thread (extends Thread)

INTERRUPT() ➔ DONT DO IT;
WATCHDOG ➔ YES SIR;

Ein Thread/Runable (muss) die Methode run() überschreiben. Ein wird dann per thread.start() gestartet

Man kann Threads synchronisieren indem man mit dem Schlüsselwort synchronized kritische Abschnitte definiert. Diese kritischen Abschnitte werden blockiert sobald ein Thread in ihm agiert und lässt andere währenddessen warten.

⇨ Die Threads können sich nicht in ihre Arbeit reinpfuschen

COMMONS CLI

Beispiel:

```
Option lager = OptionBuilder.withArgName("lager_dir").hasArg().
        withDescription("the storage directory").withLongOpt("lager").create('l');
```

Zuerst erstellen wir eine Option mithilfe des OptionBuilder (näheres dazu siehe Quellen)

```
Options options = new Options();
options.addOption(log);
```

Nun erstellen wir ein neues Option**s** Objekt und fügen die erstellten Option Objekte hinzu.

```
CommandLineParser cmd = new BasicParser();
try {
    CommandLine line = cmd.parse(options, args);
    String v;
    if(line.hasOption('l'))
        v = line.getOptionValue('l');
}catch (ParseException ex) { //Do something }
```

Jetzt geben wir das erstelle Options Objekt und die Argumente aus der Main Methode einem CommandLineParser dieser gibt uns eine CommandLine zurück. Über die CommandLine können nun Dinge abgefragt werden wie:

Ist eine option gesetzt? hasOption(char opt)
Wie ist das Wert einer Option? getOptionValue(char opt)

Achtung: cmd.parse(Options opts, String[] args) wirft eine ParseException, diese muss gecatched werden!

Log4j

Beispiel XML-Config (log4j2.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
        <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} %level{length=5} %c{1} - %m\n"/>
    </Console>
    <File name="File" fileName="${sys:kfhuCTPdas}">
        <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss} %level{length=5} %c{1} - %m\n"/>
    </File>
  </Appenders>
  <Loggers>
    <Root level="trace">
      <AppenderRef ref="Console"/>
      <AppenderRef ref="File"/>
    </Root>
  </Loggers>
</Configuration>
```

In dieser Config werden zwei Appenders definiert einen auf die Console und einer in ein dynamisches File. Diese Appenders werden dann zum Root Logger hinzugefügt das heißt sie sind auch für alle Child Logger gültig sofern nicht anders definiert. Der Root Logger ist außerdem so eingestellt das er alles loggt ab einem Log-Level von trace oder höher (mehr dazu siehe Qullen).

Für die PatternLayout Formatter siehe
http://logging.apache.org/log4j/2.0/manual/layouts.html - PatternLayout

Was bedeutet nun eigentlich dieser komischer fileName?
Das ist ein kleiner Trick der nötig ist damit man den Ort des Files in welches Log4J speichert zur Laufzeit des Programms setzen kann. Das Schlüsselwort hierfür lautet **System Properties**.

```
System.setProperty("kfhuCTPdas", logDir);
File dir = new File("${sys:kfhuCTPdas}");
dir.delete();
```

Hiermit setze ich die System Property kfhuCTPdas (dieser Name ist deshalb so komisch, damit nicht zufällig, ein anderes Programm den gleichen Namen für eine System Property wählt) auf die Variable logDir. Das Problem ist das Log4J zu diesem Zeitpunkt bereits ein File mit dem Namen ${sys:kfhuCTPdas} erstellt hat dieses müssen wir nun löschen. Log4J wird nun ein neues File in dem angeben Directory und Namen erstellen.

JUnit

Wie man mehrere JUnit Klassen auf einmal auführt damit einem die Coverage richtig angezeigt wird.

Man Benötigt dazu eine Klasse die keinen Inhalt (**kann** sie natürlich haben) hat außer:

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({TestAssembler.class,TestEmployee.class,TestStorageguy.class,TestSupplier.class,TestFactory.class})
public class RunTests {


}
```

In @SuiteClasses werden einfach alle JUnit-Klassen angeführt die ausgeführt werden sollen

6. Quellen

- Apache Commons-CLI
  - http://commons.apache.org/sandbox/commons-cli2/manual/index.html
  - http://commons.apache.org/proper/commons-cli/usage.html
- Apache
  - http://logging.apache.org/log4j/2.0/manual/configuration.html
- CSV
  - http://opencsv.sourceforge.net/apidocs/index.html