Государственное образовательное учреждение

высшего профессионального образования

«Уфимский Государственный Авиационный Технический Университет»

Кафедра «Вычислительной математики и кибернетики»

Отчет

По лабораторной работе

по дисциплине: Компьютерное моделирование

Лабораторная работа № 2.

Моделирование работы участка цеха

Проверил

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил:

студент гр. ПРО-301в

Доронин С.Г.

Вариант 3

Уфа 2014

# Задание

Необходимо промоделировать работу участка цеха, состоящего из нескольких станков и обрабатывающего два потока деталей различного типа.

Маршрут обработки деталей:

1-А1 2-А2 3-А3 1-А1 2-А2 3-А3

Интервалы времени поступления деталей первого типа (мин.) 15 + 5

Интервалы времени поступления деталей второго типа (мин.) 35 + 8

Интервал времени выполнения операций (мин.)

3 1-(18 + 3) 2-(10 + 3) 3-(12 + 5) 4-(20 + 4) 5-(25 + 8) 6-(12 + 4)

# Структура решения (этапы решения и их взаимосвязь)

Для решения этой задачи были созданы 7 классов:

1. MainForm – форма для просмотра статистики работы цеха
2. Workshop – класс, отвечающий за работу цеха
3. MachineTool – класс, отвечающий за работу станков
4. DetailGenerator – класс, отвечающий за поставку деталей
5. Detail – класс детали
6. DetailType – тип детали
7. Operation – класс

Программа запускается с класса **MainForm**, далее создается цех (**Workshop**) и 4 станка (**MachineTool**). **Workshop** создает 2 поставщика деталей (**DetailGenerator**) (для каждого типа деталей свой поставщик). **DetailGenerator** создает деталь (**Detail**) определенного типа (**DetailType**), тип детали хранит информацию о том, какие операции должны выполняться над деталью (**Opertaion**).

# Описание интерфейса с пользователем (руководство пользователя)

Программа запускается с файла **KM-assembly-0.1-SNAPSHOT.jar** или с файла сценария **run.bat.** Для работы приложения требуется установленная JRE 1.7 (Java Runtime Environment).

После запуска программы появится окно, имеющее вид (Рисунок 1).

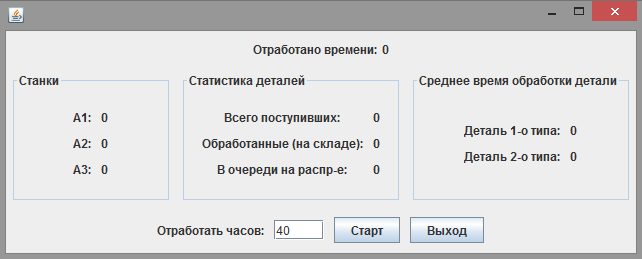


Рисунок 1. Главное окно программы

Для того чтобы начать работу цеха, нужно ввести количество, часов которые он должен отработать в поле: «Отработать часов» и нажать кнопку «Старт».

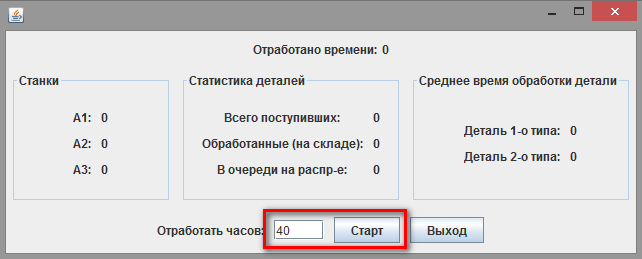


Рисунок 2. Старт работы цеха

Цех начнет работать и цифры в статистике будут менять свои значения, в самом верху в поле «Отработано времени» будет показано количество часов, которые цех уже отработал (Рисунок 3).

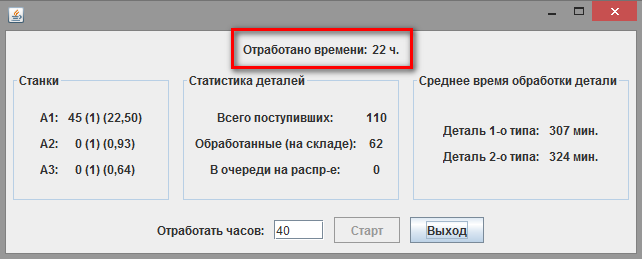


Рисунок 3. Цех в работе

В группе «Станки» отображаются 3 станка: A1, A2, A3, далее число показывает сколько сейчас деталей в очереди на этот станок, далее в скобках, число обрабатываемых в данный момент деталей на станке и в конце в скобках – средняя загрузка станка (Рисунок 4).

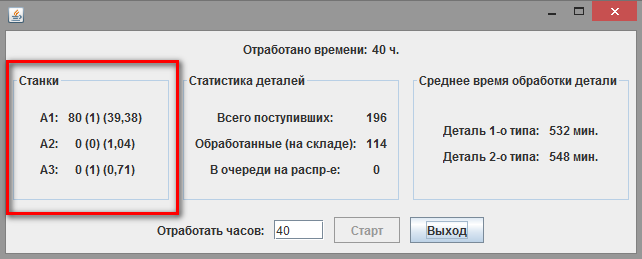


Рисунок 4. Статистика работы станков

В группе «Статистика деталей», отображается количество всего поступивших деталей всех видов, сколько деталей полностью обработаны и поступили на склад, сколько деталей находятся в очереди на распределение по станкам или на склад (Рисунок 5).

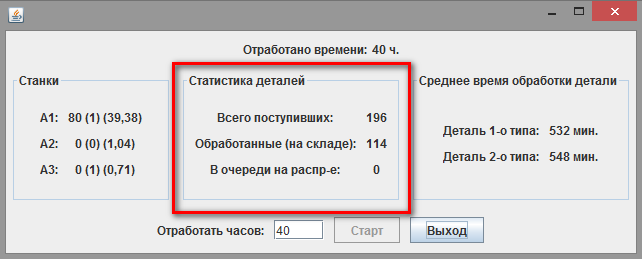


Рисунок 5. Статистика деталей

В группе «Среднее время обработки детали» отображается среднее время обработки деталей каждого типа (Рисунок 6).

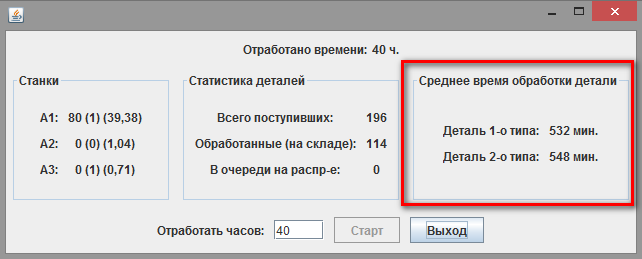


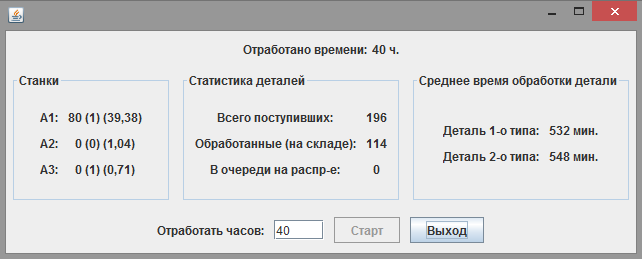
Рисунок 6. Среднее время обработки детали

Для завершения программы достаточно нажать на кнопку «Выход» или на крестик в правом верхнем углу главного окна.

# Результаты

Таблица 1. Результаты работы цеха

|  |  |
| --- | --- |
| **Показатель** | **Значение** |
| Средняя загрузка 1-о станка | 39.38 |
| Средняя загрузка 2-о станка | 1.04 |
| Средняя загрузка 3-о станка | 0.71 |
| Среднее время обработки деталей 1-о типа (мин) | 532 |
| Среднее время обработки деталей 2-о типа (мин) | 548 |
| Очередь на 1-ом станке | 80 |
| Очередь на 2-ом станке | 0 |
| Очередь на 3-ом станке | 0 |
| Поступило деталей | 196 |
| Полностью обработано деталей | 114 |



# Код программы

## Цех

/\*\*

\* Мастерская

\*/

**class** Workshop(workHours: Int) **extends** Thread {

/\*\* Состояние работы мастерской \*/

**val** enable **=** **new** AtomicBoolean(true)

/\*\* Очередь необработанных деталей \*/

**val** detailQueue **=** mutable.Queue[Detail]()

/\*\* Склад обработанных деталей \*/

**val** warehouse **=** ListBuffer[Detail]()

/\*\* Logger \*/

**private** **val** logger **=** **new** Logger(true, true, false)

/\*\* Всего сгенерировано деталей \*/

**val** generateDetailCount **=** **new** AtomicInteger(0)

/\*\* Добавление детали в очередь \*/

**def** addDetailToQueue(detail: Detail, isGenerate: Boolean): Unit **=** synchronized {

logger.debug(s"Add detail to queue: $detail")

**if** (isGenerate) generateDetailCount.getAndAdd(1)

detailQueue **+=** detail

}

// Поток деталей 1-го типа

**private** **val** incomingIntervalTypeOne **=** Timeslot(15, 5)

**private** **val** generatorV1 **=** **new** DetailGenerator(DetailType.V1, addDetailToQueue, incomingIntervalTypeOne)

// Поток деталей 2-го типа

**private** **val** incomingIntervalTypeTwo **=** Timeslot(35, 8)

**private** **val** generatorV2 **=** **new** DetailGenerator(DetailType.V2, addDetailToQueue, incomingIntervalTypeTwo)

/\*\* Время начало работы \*/

**private** **var** startDate: Date **=** **null**

/\*\* Отработанное время (мс.) \*/

**def** workedTime **=** **new** Date().getTime **-** startDate.getTime

/\*\* Запуск поставки и обработки деталей \*/

**override** **def** run() {

startDate **=** **new** Date()

// Запуск поставки деталей

generatorV1.start()

generatorV2.start()

// Запуск станков

MachineTool.setAction(addDetailToQueue)

MachineTool.startAll()

// Размещение деталей по станкам

while(enable.get) {

detailQueue.synchronized {

**if** (detailQueue.nonEmpty) {

**val** detail **=** detailQueue.dequeue()

**val** currentOperation **=** detail.currentOperation

// Если есть следующая операция, то помещаем деталь в очередь станка, выполняющего эту операцию

**if** (currentOperation.isDefined) {

**val** machineTool **=** currentOperation.get.machineTool

logger.debug(s"Detail: $detail add to $machineTool")

machineTool.addDetail(detail)

}

// Если нет, значит деталь обработана полностью, отправляем ее на склад

**else** addToWarehouse(detail)

}

}

Thread.sleep(Const.ThreadSleepMilis)

}

}

// Время работы мастерской

**private** **val** endDate **=** **new** Date(new Date().getTime **+** (workHours \* 60 \* 60 \* 1000 / Const.Acceleration))

**new** Timer().schedule(new TimerTask {

**override** **def** run() {

logger.debug("Switch off workshop")

stopGenerateDetail()

MachineTool.stopAll()

enable.set(false)

}

}, endDate)

/\*\* Остановка поставки деталей \*/

**def** stopGenerateDetail() {

generatorV1.stop()

generatorV2.stop()

}

/\*\* Добавить деталь на склад \*/

**private** **def** addToWarehouse(detail: Detail) {

logger.debug(s"Add detail to warehouse: $detail")

detail.calcHandlerTime()

warehouse **+=** detail

}

/\*\* Размеры очередей на станках \*/

**def** machineToolDetailQueueSize **=** MachineTool.detailQueueSize

/\*\* Среднее время обработки деталей каждого типа (мс.) \*/

**def** avgDetailHandlerTime **=** {

**val** v1 **=** avgDetailHandlerTimeByType(DetailType.V1)

**val** v2 **=** avgDetailHandlerTimeByType(DetailType.V2)

(v1.toLong, v2.toLong)

}

/\*\* Среднее время обработки деталей по типу (мс.) \*/

**private** **def** avgDetailHandlerTimeByType(detailType: DetailType) **=** {

**val** handlerTimes **=** warehouse.filter(\_.detailType **==** detailType).map(\_.handlerTime)

handlerTimes.sum.toDouble / handlerTimes.size

}

}

## Станок

/\*\*

\* Станок

\*/

**class** MachineTool(name: String) {

/\*\* Включен или выключен станок \*/

**private** **val** enable **=** **new** AtomicBoolean(false)

/\*\* Очередь деталей \*/

**private** **val** detailQueue **=** mutable.Queue[Detail]()

/\*\* Действие над обработанной деталью \*/

**private** **var** action: (Detail, Boolean) **=>** Unit **=** **null**

/\*\* Logger \*/

**private** **val** logger **=** **new** Logger(true, true, true)

/\*\* Состояние занятости станка \*/

**private** **val** busyState **=** **new** AtomicBoolean(false)

**private** **def** buzyStateInt **=** **if** (busyState.get) 1 **else** 0

/\*\* Обработчик деталей \*/

**private** **val** handler **=** **new** Thread(new Runnable {

**override** **def** run() {

**while** (enable.get) {

detailQueue.synchronized {

**if** (detailQueue.nonEmpty **&&** **!**busyState.get) {

busyState.set(true)

**val** currentDetail **=** detailQueue.dequeue()

logger.debug(s"${MachineTool.this} processes $currentDetail")

currentDetail.operation { v **=>** busyState.set(false) }

action(currentDetail, false)

}

}

Thread.sleep(Const.ThreadSleepMilis)

}

}

})

/\*\* Средняя загрузка станка \*/

**private** **val** avgLoad **=** **new** Stat

//--------------------------------------------------

// Подсчет средней загрузки станка

**new** Timer().schedule(new TimerTask() {

**override** **def** run() {

avgLoad.newElement()

avgLoad.add(detailQueue.size **+** buzyStateInt)

}

}, 0, 1000)

/\*\* Добавление детали в очередь \*/

**def** addDetail(detail: Detail) **=** synchronized {

logger.debug(s"$this add $detail")

detailQueue **+=** detail

}

/\*\* Запуск станка \*/

**def** start() {

logger.debug(s"Start $this")

enable.set(true)

handler.start()

}

/\*\* Остановка станка \*/

**def** stop() {

logger.debug(s"Stop $this")

enable.set(false)

}

**override** **def** toString **=** s"MachineTool($name)"

}

/\*\*

\* Станки участка цеха

\*/

**object** MachineTool {

**def** apply(name: String) **=** **new** MachineTool(name)

**val** A1 **=** MachineTool("A1")

**val** A2 **=** MachineTool("A2")

**val** A3 **=** MachineTool("A3")

**private** **val** list **=** List(A1, A2, A3)

/\*\* Установить действие над обработанной деталью всем станкам \*/

**def** setAction(action: (Detail, Boolean) **=>** Unit) **=** list.foreach(\_.action **=** action)

/\*\* Запустить все станки \*/

**def** startAll() **=** list.foreach(\_.start())

/\*\* Остановить все станки \*/

**def** stopAll() **=** list.foreach(\_.stop())

/\*\* Размеры очередей на станках \*/

**def** detailQueueSize **=** list.map(e **=>** (e.detailQueue.size, e.buzyStateInt, e.avgLoad.avg))

}

## Поставщик деталей

/\*\*

\* Генератор деталей

\*/

**class** DetailGenerator(detailType: DetailType, action: (Detail, Boolean) **=>** Unit, incomingInterval: Timeslot) {

/\*\* Флаг работы генератора \*/

**private** **val** enable **=** **new** AtomicBoolean(false)

/\*\* Logger \*/

**private** **val** logger **=** **new** Logger(true, true, true)

/\*\* Поток для генерации деталей \*/

**private** **val** generator **=** **new** Thread(new Runnable {

**override** **def** run() {

**val** detailName **=** "Detail"

**var** detailIndex **=** 0

**while** (enable.get) {

detailIndex **+=** 1

**val** incomingMilis **=** incomingInterval.get

logger.debug(s"Incoming interval: ${incomingMilis/1000} s.")

Thread.sleep(incomingMilis)

**val** detail **=** Detail(s"$detailName-$detailIndex", detailType)

logger.debug(s"DetailGenerator generate detail: $detail")

action(detail, true)

}

}

})

/\*\* Генерация деталей \*/

**def** start() {

logger.debug("DetailGenerator start")

enable.set(true)

generator.start()

}

/\*\* Остновка генерации деталей \*/

**def** stop() {

enable.set(false)

}

}