**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ**

**ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ**

**им. проф. М.А. Бонч-Бруевича»**

**(СПбГУТ)**

**Санкт-Петербургский колледж телекоммуникаций**

**Отчет о выполнении  
практического занятия №09-10**

Выполнил: Обучающиеся 4 курса, 581 группы,

Филипович Валерий Анатольевич

Состав бригады: -

Проверил: преподаватель  
Баталов Дмитрий Иннокентьевич

Санкт-Петербург

2021 г.

**Постановка задачи**

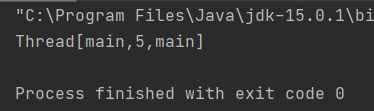
Реализовать два консольных приложения со следующими условиями:

* **Задание 1.** Запустить на выполнение код программ, приведенных в конспекте лекционного занятия «Занятие № 9. Многопоточное программирование». Выполнять только те программы, для которых приведены примеры вывода на консоль. Обращать внимание на возникающие ошибки компиляции, которые нужно исправить. Добиться успешного выполнения программ.
* **Задание 2.** В качестве примера рассмотреть программы построения апплетов с применением многопоточности. Разобрать примеры программ, приведенных в файле «МДК.03.02\_ИСРПО\_Тема2.1\_ПЗ\_9-10», добиться понимания всех строк кода, выполнить их в среде разработки Java. Добиться успешного выполнения примеров.

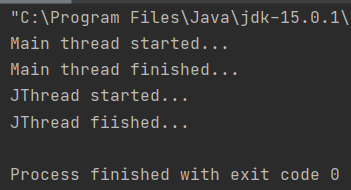
**Ход выполнения работы**

1. Создал новый консольный проект «ThreadProgram»;
2. Добавил в код фрагмент, в котором продемонстрировано получение всей информации о текущем потоке;
3. Создал новый консольный проект «ThreadExtendsProgram»;
4. Добавил в код фрагмент, в котором продемонстрировано наследование от класса Thread;
5. Создал новый консольный проект «ManyThreadsProgram»;
6. Добавил в код фрагмент, в котором продемонстрировано наследование от класса Thread и создание сразу нескольких потоков;
7. Создал новый консольный проект «ThreadJoinProgram»;
8. Добавил в код фрагмент, в котором продемонстрирована функция ожидания завершения потока;
9. Создал новый консольный проект «ThreadRunnableProgram»;
10. Добавил в код фрагмент, в котором продемонстрирована реализация интерфейса Runnable;
11. Создал новый консольный проект «ThreadInterruptProgram»;
12. Добавил в код фрагмент, в котором реализован метод interrupt();
13. Создал новый консольный проект «ThreadSynchronizedProgram»;
14. Добавил в код фрагмент, в котором продемонстрирован общий ресурс потоков;
15. Создал новый консольный проект «ThreadSynchronizedSecondProgram»;
16. Добавил в код фрагмент, в котором продемонстрировано использование ключевого слова synchronized;
17. Создал новый консольный проект «ThreadWaitNotifyProgram»;
18. Добавил в код фрагмент, в котором продемонстрировано использование методов wait и notify;
19. Создал новый консольный проект «ThreadSemaphoreProgram»;
20. Добавил в код фрагмент, в котором продемонстрировано использование семафора;
21. Создал новый консольный проект «ThreadPhilosopherProgram»;
22. Добавил в код фрагмент, в котором продемонстрировано использование семафора;
23. Создал новый консольный проект «ThreadExchangerProgram»;
24. Добавил в код фрагмент, в котором продемонстрировано использование класса Exchanger;
25. Создал новый консольный проект «ThreadPhaserProgram»;
26. Добавил в код фрагмент, в котором продемонстрировано использование класса Phaser;
27. Создал новый консольный проект «ThreadPhaserSleepProgram»;
28. Добавил в код фрагмент, в котором продемонстрировано использование класса Phaser и метод sleep;
29. Создал новый консольный проект «MultiThreadsProgram»;
30. Добавил в код фрагмент, в котором разработана процедура создания и запуска потоков на примере использования их в апплетах;
31. Создал новый консольный проект «AnimationProgram»;
32. Добавил в код фрагмент, в котором продемонстрировано применение анимации для мультизадачности.

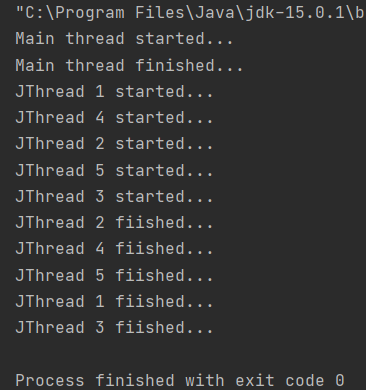
**Результат выполнения программы**



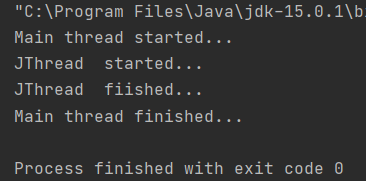
*Рис. 1. Результат работы программы «ThreadProgram»*



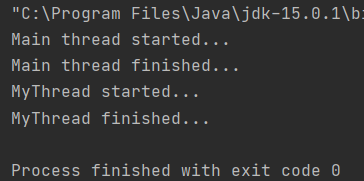
*Рис. 2. Результат работы программы «ThreadExtendsProgram»*



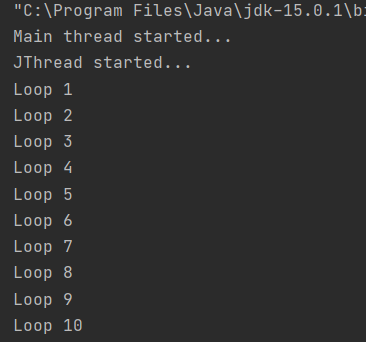
*Рис. 3. Результат работы программы «ManyThreadsProgram»*



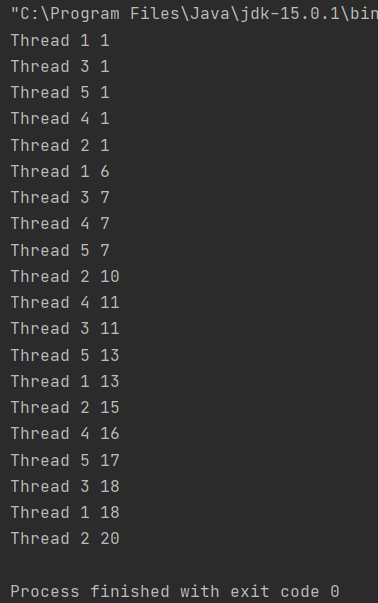
*Рис. 4. Результат работы программы «ThreadJoinProgram»*



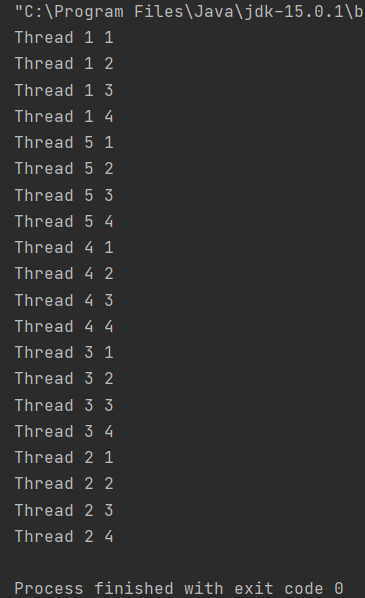
*Рис. 5. Результат работы программы «ThreadRunnableProgram»*



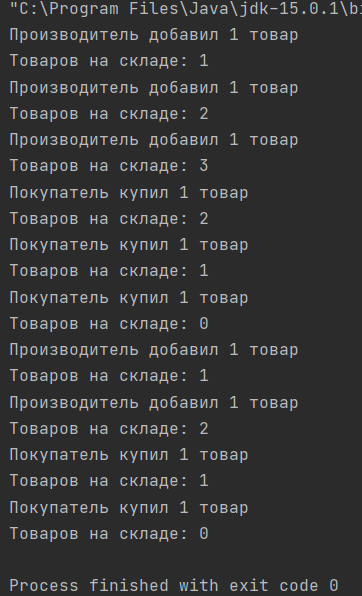
*Рис. 6. Результат работы программы «ThreadInterruptProgram»*



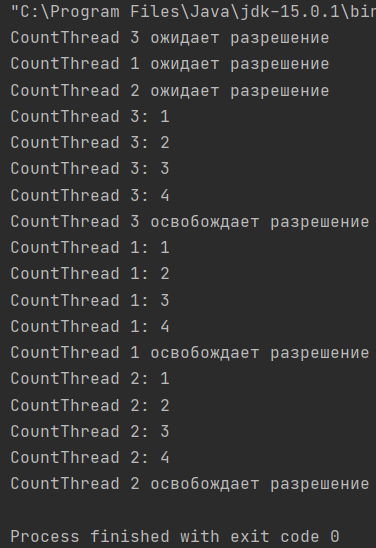
*Рис. 7. Результат работы программы «ThreadSynchronizedProgram»*



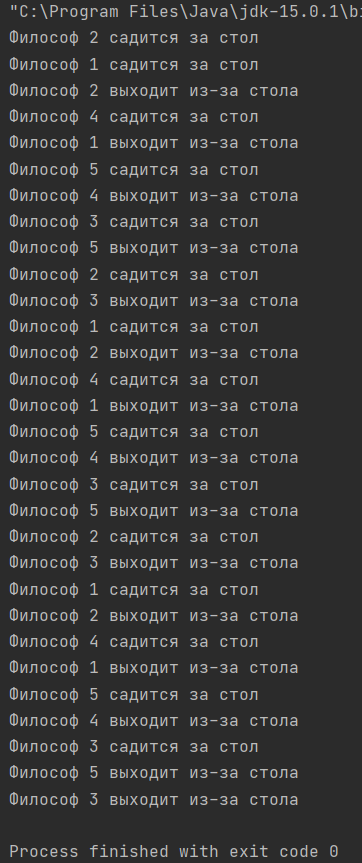
*Рис. 8. Результат работы программы «ThreadSynchronizedSecondProgram»*



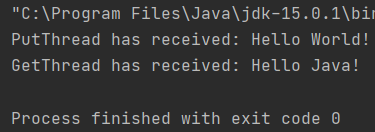
*Рис. 9. Результат работы программы «ThreadWaitNotifyProgram»*



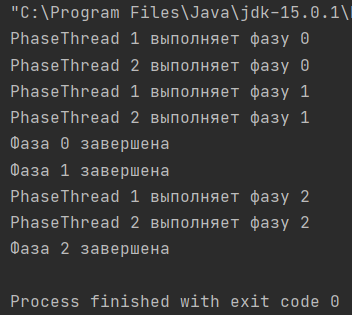
*Рис. 10. Результат работы программы «ThreadSemaphoreProgram»*



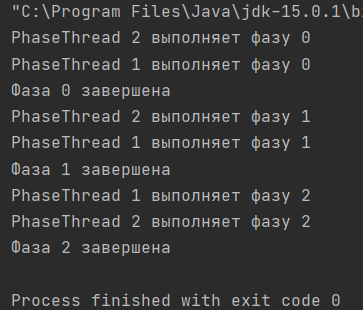
*Рис. 11. Результат работы программы «ThreadPhilosopherProgram»*



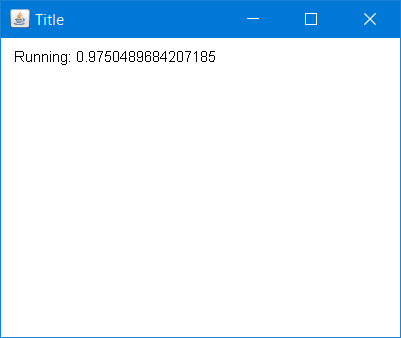
*Рис. 12. Результат работы программы «ThreadExchangerProgram»*



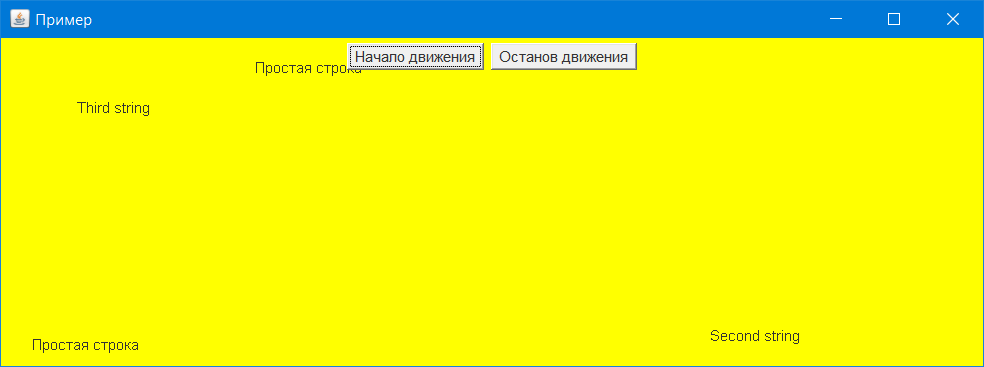
*Рис. 13. Результат работы программы «ThreadPhaserProgram»*



*Рис. 14. Результат работы программы «ThreadPhaserSleepProgram»*



*Рис. 15. Результат работы программы «MultiThreadsProgram»*



*Рис. 16. Результат работы программы «AnimationProgram»*

**Ссылка на GitHub**

<https://github.com/ValeriyFilipovich/SPbCT_FilipovichVA/tree/master/Practice_tasks/>

**Ответы на контрольные вопросы**

1. *Что такое процесс (задача) и поток (нить)?*

Процесс (задача) (process) — это объект, который создается операционной системой, когда пользователь запускает приложение.

Для каждого процесса операционная система создает один главный поток (нить) (thread), который является потоком выполняющихся по очереди команд центрального процессора.

1. *Чем определяется порядок передачи управления потокам?*

В языке Java каждый поток обладает приоритетом, который оказывает влияние на порядок его выполнения. Потоки с высоким приоритетом выполняются до потоков с низким приоритетом. Порядок их выполнения определен основной операционной системой и аппаратными средствами. Операционная система обслуживает потоки при помощи планировщика, который и определяет порядок выполнения.

1. *Какие есть способы реализации многозадачности в Java?*

Во-первых, можно создать собственный класс на базе класса Thread. При этом необходимо переопределить метод run(). Новая реализация этого метода будет работать в рамках отдельного потока.

Во-вторых, создаваемый класс, не являясь подклассом класса Thread, может реализовать интерфейс Runnable. При этом в рамках этого класса необходимо определить метод run(), который будет работать как отдельный поток.

1. *Что необходимо сделать для создания подкласса потоков (подкласса Thread)?*

Необходимо создать отдельный подкласс с наследованием от класса Thread.

1. *Когда запускается на выполнение метод run() подкласса Thread?*

Метод run() не вызывается напрямую никакими другими методами. Он получает управление при запуске потока методом start() класса Thread. В случае апплетов создание и запуск потоков обычно осуществляется в методе start() апплета.

1. *Какими методами класса Thread необходимо запускать поток на выполнение и останавливать его?*

Создание и запуск потоков обычно осуществляется в методе start() апплета.

Обычно остановка всех работающих потоков, созданных апплетом, выполняется в методе stop() апплета. Сейчас использование этого метода является устаревшим и не рекомендуется. Завершение работы потока желательно проводить так, чтобы происходило естественное завершение метода run. Для этого используется управляющая переменная в потоке. В наших потоках это переменная булевского типа – going.

1. *Что необходимо сделать для реализации классом интерфейса Runnable?*

Реализация интерфейса Runnable во многом аналогична переопределению класса Thread. В методе run() собственно определяется весь тот код, который выполняется при запуске потока. После определения объекта Runnable он передается в один из конструкторов класса Thread.

1. *В каких состояниях может находиться поток?*

* Новый поток
* Выполняемый поток
* Невыполняемый поток
* Завершенный поток

1. *Какой поток считается новым, выполняемым и завершенным?*

Новый поток - при создании экземпляра потока этот поток приобретает состояние “Новый поток”: Thread myThread=new Thread(); В этот момент для данного потока распределяются системные ресурсы; это всего лишь пустой объект. В результате все, что с ним можно делать — это запустить: myThread.start(); Любой другой метод потока в таком состоянии вызвать нельзя, это приведет к возникновению исключительной ситуации.

Выполняемый поток - когда поток получает метод start(), он переходит в состояние “Выполняемый поток”. Процессор разделяет время между всеми выполняемыми потоками согласно их приоритета

Завершенный поток - когда метод run() завершается, поток переходит в состояние “Завершенный поток”.

1. *В каких ситуациях поток является невыполняемым?*

Невыполняемый поток - если поток не находится в состоянии “Выполняемый поток”, то он может оказаться в состоянии “Невыполняемый поток”. Это состояние наступает тогда, когда выполняется одно из четырех условий:

* Поток был приостановлен
* Поток ожидает.
* Поток ожидает извещения.
* Поток заблокирован другим потоком.

1. *Когда возникают исключительные ситуации при работе с потоками?*

Исполняющая система языка Java будет возбуждать исключительную ситуацию IllegalThreadStateException всякий раз, когда будет вызываться метод, которым поток не может оперировать в своем текущем состоянии.

1. *Что такое приоритетная система и какой системой является виртуальная машина Java?*

Приоритетные системы гарантируют, что в любое время будет выполняться поток с самым высоким приоритетом. Виртуальная машина Java является приоритетной, то есть выполняться всегда будет поток с самым высоким приоритетом.

1. *Что такое группы потоков и чем они полезны?*

Все потоки в языке Java должны входить в состав группы потоков. В классе Thread имеется три конструктора, которые дают возможность указывать, в состав какой группы должен входить данный создаваемый поток.

Группы потоков особенно полезны, поскольку внутри их можно запустить или приостановить все потоки, а это значит, что при этом не потребуется иметь дело с каждым потоком отдельно. Группы потоков предоставляют общий способ одновременной работы с рядом потоков, что позволяет значительно сэкономить время и усилия, затрачиваемые на работу с каждым потоком в отдельности.

1. *Что такое родовая группа потоков и главная группа потоков?*

genericGroup (родовая группа) - когда группа создана, создаются несколько потоков, входящих в ее состав.

Если при создании нового потока не указать, к какой конкретной группе он принадлежит, этот поток войдет в состав группы потоков main (главная группа) языка Java. Иногда ее еще называют текущей группой потоков.

1. *Чем определяется уровень защиты группы потоков?*

По умолчанию создаваемым потокам не присваивается определенный уровень защиты. В результате любой поток из любой группы может свободно контролировать и изменять потоки в других группах. Однако можно использовать абстрактный класс SecurityManager для указания ограничений доступа к определенным группам потоков. Для этого необходимо создать подкласс класса SecurityManager и заменить те методы, которые используются для защиты потоков. В основном эта процедура применима для потоков приложений, так как некоторые WWW-браузеры не позволяют заменить уровни защиты.

1. *Как уменьшить мерцание экрана при анимации?*

Метод update следует переопределить, чтобы не было мерцания при заполнении экрана фоном.

1. *Что такое двойная буферизация?*

При двойной буферизации создается изображение в памяти, а затем оно выводится на экран.