**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,

Лепов Алексей Валерьевич

Проверил: преподаватель,  
Баталов Дмитрий Иннокентьевич

Санкт-Петербург

2021 год

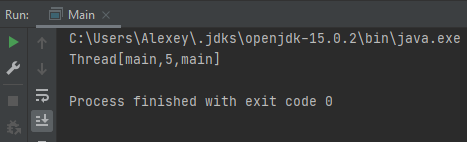
1. **Постановка задач.**

* Изучить работу с потоками.
* Посмотреть на результаты компиляции.
* Также нужно проверить свои навыки загрузки кода программ на хостинге проектов программного обеспечения github.com

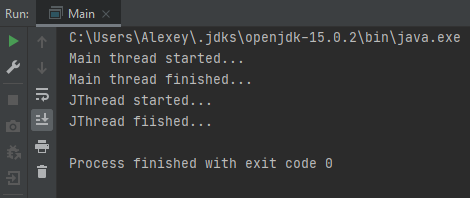
1. **Ход выполнения работ.**

* Выполнено Задание 1 практической работы. Запущены на выполнение коды программ, приведенных в конспекте лекционного занятия «Занятие № 9. Многопоточное программирование».
* Выполнено Задание 2. Рассмотрены программы построения апплетов с применением многопоточности.
* Даны ответы на контрольные вопросы.

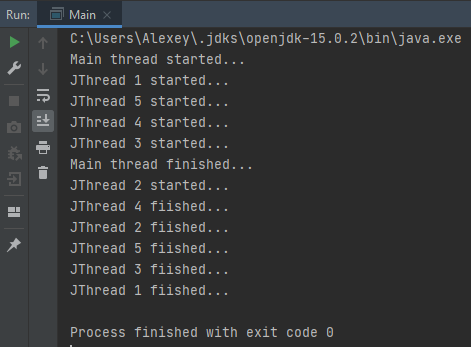
1. **Результаты компиляции программного кода.**



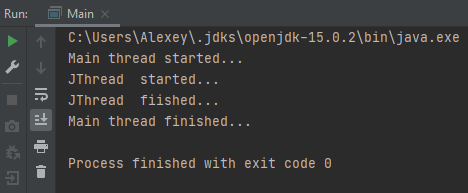
1. *Результат работы программы.* *Задание 1.*



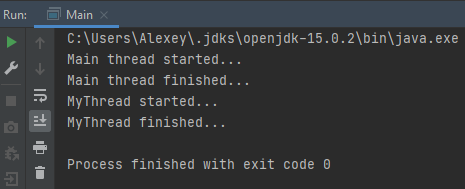
1. *Результат работы программы.* *Задание 1.*



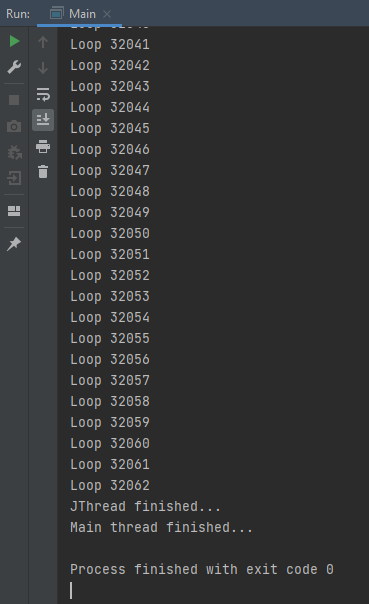
1. *Результат работы программы.* *Задание 1.*



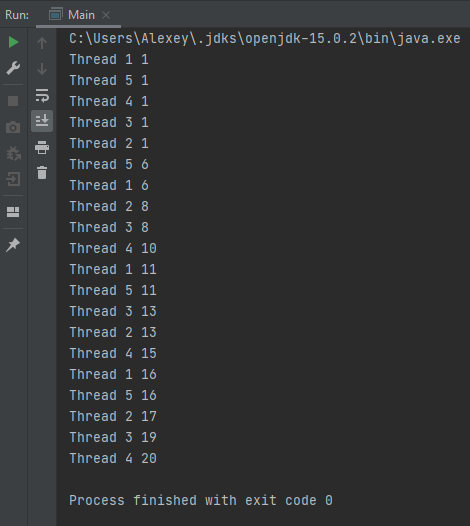
1. *Результат работы программы.* *Задание 1.*



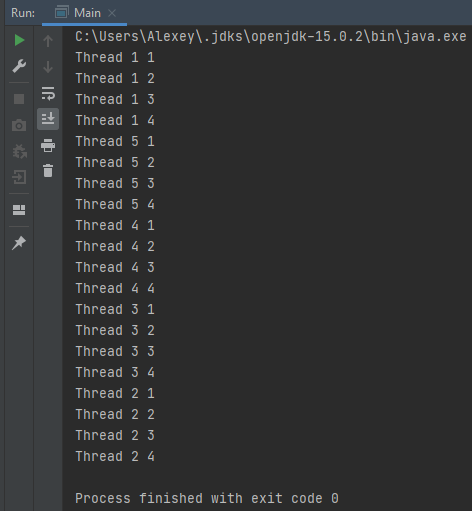
1. *Результат работы программы.* *Задание 1.*



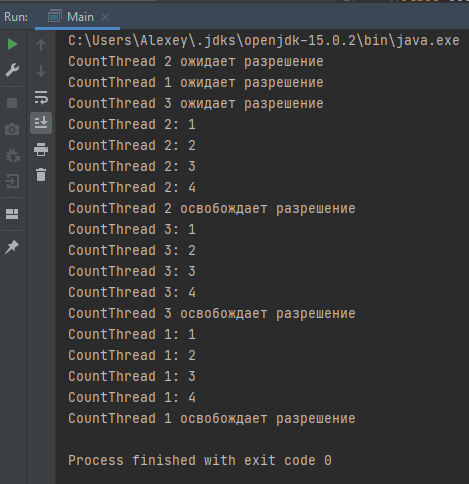
1. *Результат работы программы.* *Задание 1.*



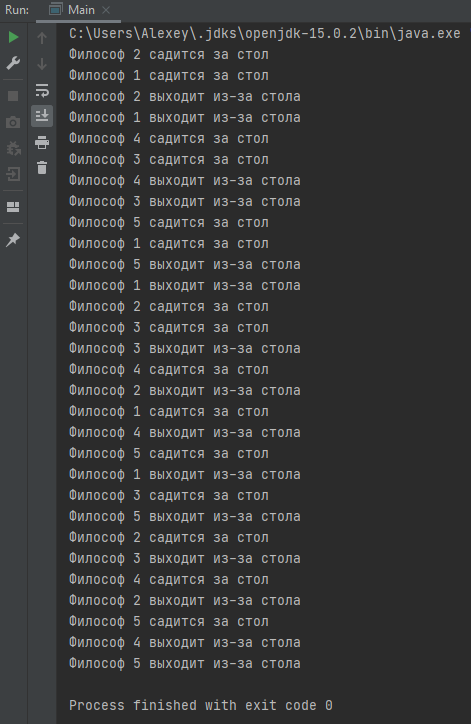
1. *Результат работы программы.* *Задание 1.*



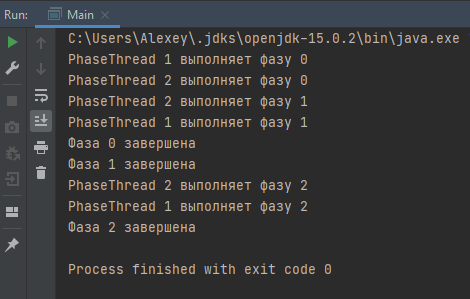
1. *Результат работы программы.* *Задание 1.*



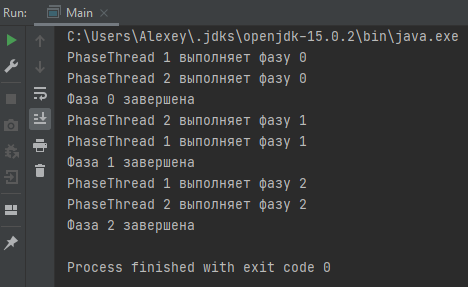
1. *Результат работы программы.* *Задание 1.*



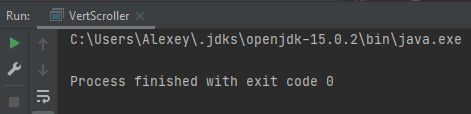
1. *Результат работы программы.* *Задание 1.*



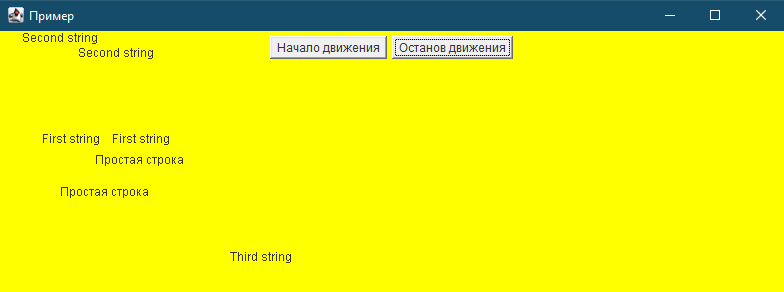
1. *Результат работы программы.* *Задание 1.*



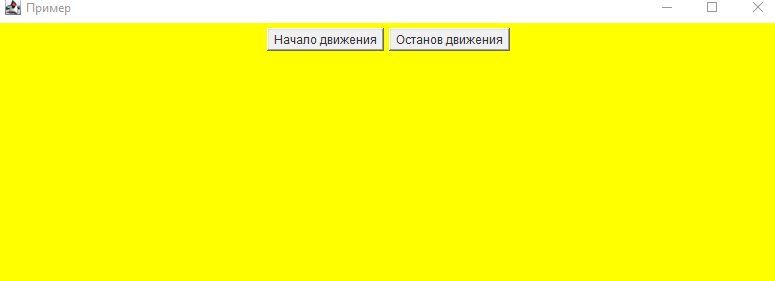
1. *Результат работы программы.* *Задание 1.*



1. *Результат работы программы.* *Задание 2.*



1. *Результат работы программы.* *Задание 2.*

**

1. *Результат работы программы.* *Задание 2.*
2. **Исходный код проектов.**

Выгрузка проекта была реализована с помощью онлайн инструментария GitHub: (<https://github.com/AlexeyLepov/SPbCK_Lepov_PZ_09-10.git>).

1. **Вопросы для защиты.**
2. ***Что такое процесс (задача) и поток (нить)?***

**Процесс** или **задача**– это обработка операционной системой запущенного исполняемого кода, который в свою очередь может породить и другие процессы. Процессу выделяется отдельное адресное пространство, причем это пространство физически недоступно для других процессов. Процесс может работать с файлами или с каналами связи локальной или глобальной сети.

**Потоки** или **нити** – это параллельно выполняемые части одной программы. Для каждого процесса операционная система создает один главный поток (thread), который является потоком выполняющихся по очереди команд центрального процессора. При необходимости главный поток может создавать другие потоки, пользуясь для этого программным интерфейсом операционной системы.

1. ***Чем определяется порядок передачи управления потокам?***

В случаях, где два потока с одинаковым приоритетом конкурируют за циклы CPU, ситуация немного сложнее. Для операционных систем типа Windows 98, потоки равного приоритета квантуются (во времени) автоматически, циклическим способом. Для других типов операционных систем, типа Solaris 2.x, потоки равного приоритета должны добровольно передавать управление другим потокам. Если они этого не делают, другие потоки не будут выполняться.

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

Для создания мультизадачных приложений Java необходимо воспользоваться классом java.lang.Thread. В этом классе определены все методы, необходимые для создания потоков, управления их состоянием и синхронизации.

**Есть две возможности использования класса Thread.**

* 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 рамка этого класса необходимо определить метод run(), который будет работать как отдельный поток.

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

Для создания нового потока мы можем создать новый класс, либо наследуя его от класса Thread, либо реализуя в классе интерфейс Runnable.

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

Метод run() вызывается методом start().

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

* void start() - запуск потока.
* void wait() - приостановка потока, пока другой поток не вызовет метод notify();

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

Реализация интерфейса Runnable используется в случаях, когда класс уже наследует какой-либо родительский класс и не позволяет расширить класс Thread. К тому же хорошим тоном программирования в java считается реализация интерфейсов. Это связано с тем, что в java может наследоваться только один родительский класс. Таким образом, унаследовав класс Thread, невозможно наследовать какой-либо другой класс.

1. ***В каких состояниях может находиться поток?***

Поток может находиться в одном из следующих состояний:

* ***New*** - объект класса *Thread* создан, но еще не запущен. Он еще не является потоком выполнения и, естественно, не выполняется.
* ***Runnable*** - поток готов к выполнению, но планировщик еще не выбрал его.
* ***Running***– поток выполняется.
* ***Waiting/blocked/sleeping*** - поток блокирован или поток ждет окончания работы другого потока.
* ***Dead*** - поток завершен. Будет выброшено исключение при попытке вызвать метод *start()* для dead потока.

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

Поток считается новым при создании экземпляра потока. В этот момент для данного потока распределяются системные ресурсы; это всего лишь пустой объект. В результате все, что с ним можно делать - это запустить его или остановить:

* *myThread.start();*
* *myThread.stop();*

Когда поток получает метод start(), он переходит в состояние «**Выполняемый поток**». Процессор разделяет время между всеми выполняемыми потоками согласно их приоритета

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*. Это условие является результатом вызова метода suspend(). После вызова этого метода поток не находится в состоянии готовности к выполнению; его сначала нужно «разбудить» с помощью метода resume(). Это полезно в том случае, когда необходимо приостановить выполнение потока, не удаляя его.

· *Поток ожидает.* Это условие является результатом вызова метода sleep(). После вызова этого метода поток переходит в состояние ожидания в течении некоторого определенного промежутка времени и не может выполняться до истечения этого промежутка Даже если ожидающий поток имеет доступ к процессору, он его не получит. Когда указанный промежуток времени пройдет, поток переходит в состояние «Выполняемый поток». Метод resume() не может повлиять на процесс ожидания потока, этот метод применяется только для приостановленных потоков.

· *Поток ожидает извещения.* Это условие является результатом вызова методаwait(). С помощью этого метода потоку можно указать перейти в состояние ожидания выполнения условия, определяемого переменной условия, вынуждая его тем самым приостановить свое выполнение до тех пор, пока данное условие удовлетворяется. Какой бы объект не управлял ожидаемым условием, изменение состояния ожидающих потоков должно осуществляться посредством одного из двух методов этого потока - notify() или notifyAll(). Если поток ожидает наступление какого-либо события, он может продолжить свое выполнение только в случае вызова для него этих методов.

· *Поток заблокирован другим потоком*. Это условие является результатом блокировки операцией ввода-вывода или другим потоком. В этом случае у потока нет другого выбора, как ожидать до тех пор, пока не завершится команда ввода-вывода или действия другого потока. В этом случае поток считается невыполняемым, даже если он полностью готов к выполнению.

Поток считается завершенным, когда метод run() завершается. Если в методе run() содержится бесконечный цикл, то завершить работу потока можно при помощи вызова его метода stop().

1. ***В каких ситуациях поток является невыполняемым?***

Поток заблокирован другим потоком. Это условие является результатом блокировки операцией ввода-вывода или другим потоком. В этом случае у потока нет другого выбора, как ожидать до тех пор, пока не завершится команда ввода-вывода или действия другого потока. В этом случае поток считается невыполняемым, даже если он полностью готов к выполнению.

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

После того как программа запущена на выполнение в операционную систему, подстерегает множество неожиданностей и опасностей. Например, она не сможет прочитать необходимый файл, так как испорчен сектор на диске, или вообще не найдет файл с таким именем. В компьютере может не хватить оперативной памяти для ее скромных запросов и т.д. Такие ситуации в языке Java называют исключительные ситуации или исключения.

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

Система, которая имеет дело со множеством выполняющихся потоков, может быть или приоритетная, или неприоритетная. 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.

Сама виртуальная машина генерирует хеш-код, основываясь на расположении объекта в памяти. Это позволяет для различных объектов (неравенство по ссылке) получать различные хеш-коды.

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.

Для того, чтобы отдельный поток не мог начать останавливать и прерывать все потоки подряд, введено понятие группы. Поток может оказывать влияние только на потоки, которые находятся в одной с ним группе. Группу потоков представляет класс ThreadGroup. Такая организация позволяет защитить потоки от нежелательного внешнего воздействия.

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?***

Группа потоков может содержать другие группы, что позволяет организовать все потоки и группы в иерархическое дерево, в котором каждый объект ThreadGroup, за исключением корневого, имеет родителя.

В приведенном ниже фрагменте программы создается группа потоков под названием genericGroup (**родовая группа**). Когда группа создана, создаются несколько потоков, входящих в ее состав:

ThreadGroup genericGroup=new ThreadGroup("My generic group");

Thread t1=new Thread(genericGroup,this);

Thread t2=new Thread(genericGroup,this);

Thread t3=new Thread(genericGroup,this);

Совсем не обязательно создавать группу, в состав которой входят потоки программы. Можно воспользоваться группой, созданной исполняющей системой Java, или группой, созданной приложением, в котором выполняется апплет.

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 SecurityManagerдля 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-броузеры не позволяют заменить уровни защиты.

1. ***Как уменьшить мерцание экрана при анимации?***

Использование методов **update и paint**

Пусть некоторый объект должен перемещаться в пределах экрана. В этом случае нет необходимости полностью очищать и перерисовывать экран. Тогда в методе paintнадо закрашивать и перерисовывать только маленькую часть экрана. Для вызова методаpaintможно использовать методrepaint. Но при этом будет перерисовываться закрашиванием фоновым цветом весь экран. Чтобы экран не закрашивался фоном, необходимо переопределить методupdate. В нем надо напрямую вызвать методpaint.

Желательно, чтобы метод paintвыполнялся как можно быстрее, потому что обычно при выполнении этого метода не реагируют на события элементы управления.

1. ***Что такое двойная буферизация?***

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

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

Поддержка двойной буферизации встроена в Swing-компоненты, но для AWT-компонентов ее надо реализовывать. Ниже приводится пример апплета двойного назначения с реализацией двойной буферизации. Измененный код выделен жирным шрифтом.