Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Разработка кроссплатформенных приложений»

Инструкция

по выполнению лабораторной работы
«Разработка, отладка и испытание программ с потоками различных приоритетов»

Минск 2025 г.

Лабораторная работа № 15

Тема работы: «Разработка, отладка и испытание программ с потоками различных приоритетов»

1 Цель работы

Обучить управлению потоками посредством изменения приоритетов потоков.

2 Задание

Номер варианта соответствует вашему номеру по списку, если номер по списку больше последнего варианта, то необходимо начать подсчет варианта с начала (например, учащийся с номером по списку 13 должен выполнить 2 вариант).

В задании лабораторной работы № 14 необходимо создать несколько объектов классов, для каждого из них задать различный приоритет, запустить все потоки одновременно, и проанализировать, как приоритет патока влияет на производительность.

3 Оснащение работы

Задание по варианту, ЭВМ, среда разработки IntelliJ IDEA.

4 Основные теоретические сведения

Планировщик потоков использует **приоритеты потоков исполнения**, чтобы принять решение, когда разрешить исполнение каждому потоку. Теоретически высокоприоритетные потоки исполнения получают больше времени ЦП, чем низкоприоритетные.

А на практике количество времени ЦП, которое получает потоки исполнения, нередко зависит не только от его приоритета, но и от ряда других факторов. (Например, особенности реализации многозадачности в операционной системе могут оказывать влияние на относительную доступность времени ЦП.).

Высокоприоритетный поток исполнения может также вытеснять низкоприоритетный. Например, когда низкоприоритетный поток исполняется, а высокоприоритетный собирается возобновить свое исполнение, прерванное в связи с приостановкой или ожиданием завершения операции ввода-вывода, то он вытесняет низкоприоритетный поток.

Теоретически потоки исполнения с одинаковым приоритетом должны получать равный доступ к ЦП. Но не следует забывать, что язык Java предназначен для применения в обширном ряде сред.

В одних из этих сред многозадачность реализуется совершенно иначе, чем в других. В целях безопасности потоки исполнения с одинаковым приоритетом должны получать управление лишь время от времени. Этим

гарантируется, что все потоки получат возможность выполняться в среде операционной системы с невытесняющей многозадачностью.

Но на практике даже в средах с невытесняющей многозадачностью большинство потоков все-таки имеют шанс для исполнения, поскольку во всех потоках неизбежно возникают ситуации блокировки, например, в связи с ожиданием ввода-вывода.

Когда случается нечто подобное, исполнение заблокированного потока приостанавливается, а остальные потоки могут исполняться. Но если требуется добиться плавной работы многопоточной программы, то полагаться на случай лучше не стоит.

К тому же в некоторых видах задач весьма интенсивно используется ЦП. Потоки, исполняющие такие задачи, стремятся захватить ЦП, поэтому передавать им управление следует изредка, чтобы дать возможность выполняться другим потокам.

Чтобы установить приоритет потока исполнения, следует вызвать метод setPriority() из класса Thread. Его общая форма выглядит следующим образом:

```
final void setPriority(int уровень)
```

где аргумент уровень обозначает новый уровень приоритета для вызывающего потока исполнения.

Значение аргумента уровень должно быть в пределах от MIN_PRIORITY до MAX_PRIORITY.

В настоящее время эти значения равны соответственно 1 и 10. Чтобы возвратить потоку исполнения приоритет по умолчанию, следует указать значение NORM_PRIORITY, которое в настоящее время равно 5.

Эти приоритеты определены в классе Thread как статические завершенные (static final) переменные. Для того чтобы получить текущее значение приоритета потока исполнения, достаточно вызвать метод getPriority() из класса Thread, как показано ниже.

```
final int getPriority()
```

Разные реализации Java могуг вести себя совершенно иначе в отношении планирования потоков исполнения. Большинство несоответствий возникает при наличии потоков исполнения, опирающихся на вытесняющую многозадачность вместо совместного использования времени ЦП.

Наиболее безопасный способ получить предсказуемое межплатформенное поведение многопоточных программна Java состоит в том, чтобы использовать потоки исполнения, которые добровольно уступают управление ЦП.

5 Порядок выполнения работы

1. Выделить ключевые моменты задачи;
2. Построить алгоритм и теоритическую объектную модель решения задачи;

3. Запрограммировать полученный алгоритм и объектную модель;
4. Провести тестирование полученной программы.

6 Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____
Фамилия, инициалы учащегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Результат выполнения работы: _____
_____
_____

7. Контрольные вопросы и задания

1. Какой приоритет имеет поток по умолчанию?
2. Какие можно задать приоритеты для потока?
3. Как влияет на работу потока изменение приоритета потока?

8. Рекомендуемая литература

1. **Урванов, Ф. В.** Java. Состояние языка и его перспективы. / Ф.В. Урванов. - Санкт-Петербург : БХВ-Петербург, 2023. - 368 с.
2. **Копец Дэвид.** Классические задачи Computer Science на языке Java. - Санкт-Петербург : Питер, 2022. - 288 с
3. **Васильев А. Н.** Java. Объектно-ориентированное программирование: Учебное пособие / А.Н. Васильев. - Санкт-Петербург : Питер, 2021. - 400 с.
4. **Эванс Бенджамин**. Java для опытных разработчиков. 2-е изд. — (Серия «Библиотека программиста»). - Санкт-Петербург : Питер, 2024. - 736 с.
5. **Лой Марк.** Программируем на Java. 5-е межд. изд. . - Санкт-Петербург : Питер, 2023. - 544 с.
6. **Гетц Брайан**. Java Concurrency на практике. - Санкт-Петербург : Питер, 2021. - 464 с.