Практическая работа “Описание классов, полей, методов в Java”

Следующие несколько практических работ создают основу приложения для управления личными задачами пользователя. Пользователь может запланировать, например, «Сходить в боулинг с друзьями в среду» или «Пробегать 3 км каждый день в шесть часов утра». В рамках нашего курса мы создадим библиотеку, с помощью которой можно реализовать данное приложение. В рамки практик мы не включили создание пользовательского интерфейса, но добавили эту часть дополнительным материалом.

ОБЪЕКТЫ "ЗАДАЧА"

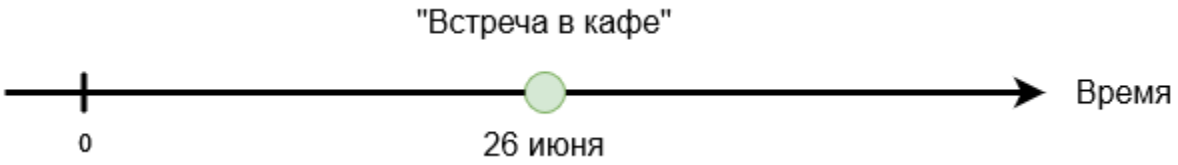
Основными объектами, с которыми будет работать приложение, являются задачи.

Задачи имеют *некоторый текст, описывающий детали задачи*, например, «Уборка помещения».

Кроме того, задачи могут быть *активными и неактивными* – например, во время отпуска утренняя пробежка может быть неактивной и временно не выполняться.

Для описания времени начала будут использоваться целые числа, которые означают количество часов, прошедших с начала отсчета. Например, если мы начнем отсчет с 1 января 2022 года, 00:00, число 36 будет означать 12:00 2 января 2022 года.

Задачи могут быть запланированы к выполнению *один раз*, например, «Встреча в кафе 26 июня в 18:00»:



Или задача может быть запланирована к *регулярному выполнению* на определенный период времени с заданным интервалом (в часах), например, «Утренняя пробежка с 1 по 4 июня каждый день в 8:00»:

Целью первой практической работы будет создание класса объектов Task.

Требования к практике

Все классы, начиная с этой практики должны принадлежать подпакетам пакета `com.nc.edu.ta.`*`yourname`*`.prN.*`, где вместо *yourname* укажите Вашу фамилию, а *N* — номер практики.

Все публичные поля и методы должны быть задокументированы с использованием JavaDoc.

Код проекта должен следовать Oracle Java Code Style Convention.

Шаг 1

Просмотрите видео «Введение в ООП и Java» и «Синтаксис Java». Создайте класс со всеми необходимыми полями и методами. Перечень необходимых методов приведен ниже. Дополните код так, чтобы он компилировался (минимально – добавьте `return`, где это нужно). На этом этапе мы делаем заготовку, потому возвращаемый результат может отличаться от желаемого. Но если какую-то часть кода получится написать сразу – это лучший вариант.

Создайте класс `com.nc.edu.ta.`*`yourname`*`.prN.Task`, обладающий следующими методами установки и получения значений полей класса:

- Методы получения и установки заголовка задачи:
  - String getTitle()
  - void setTitle(String title), где title – заголовок (название) задачи
- Методы проверки статуса задачи:
  - boolean isActive()
  - void setActive(boolean active), где active – булев индикатор активности задачи
- Методы задания и получения времени оповещения о задаче:
  - void setTime(int time) — для единоразовой задачи, где time – время оповещения о задаче. Если задача повторяется, она должна стать неповторяющейся.
  - void setTime(int start, int end, int repeat) — для повторяющейся задачи, где start – время начала оповещения о задаче, end – время прекращения оповещения о задаче , repeat – интервал времени, через который необходимо повторить оповещение о задаче
  - int getTime() — время начала оповещения (для повторяющейся задачи) или время единственного оповещения (для единоразовой задачи)
  - int getStartTime() — время начала оповещения (для повторяющейся задачи) или время единственного оповещения (для единоразовой задачи)
  - int getEndTime() — время окончания оповещения (для повторяющейся задачи) или время единственного оповещения (для единоразовой задачи)
  - int getRepeatInterval() — интервал времени, через который необходимо повторить оповещение о задаче (для повторяющейся задачи) или 0 (для единоразовой задачи)
  - boolean isRepeated() — информация о том, повторяется ли задача
- Метод, возвращающий описание данной задачи:
  - String toString() — возвращает строку со следующей информацией:

- Task “*<title>” is inactive* – для неактивных задач
- Task “*<title>” at <time>* – для активных единоразовых задач
- Task “*<title>*” from *<start>* to *<end>* every *<repeat> hours* – для активных повторяющихся задач. Пример: *Task “Сделать зарядку для зрения” from 1 to 10 every 2 hours*

- Метод, возвращающий время следующего оповещения, после указанного времени time (не включая его):
  - int nextTimeAfter(int time) — если после указанного времени оповещений больше нет или задача неактивна, то результат должен быть -1.

Используя описанные выше методы, создайте два конструктора класса Task так, чтобы новая задача при создании считалась неактивной:

- Task(String title, int time) – для единоразовой задачи
- Task(String title, int start, int end, int repeat) – для повторяющейся задачи

Шаг 2 Javadoc

После того, как каркас создан, задокументируйте все публичные поля и методы с использованием javaDoc (хорошее описание).

Шаг 3 Валидация

Просмотрите видео «Java: описание классов, модификаторы доступа, сборка мусора» и «Java: простые типы данных и операторы».

Используя материал про блоки if (ветвление) и циклы обновите код так, чтобы он реализовывал весь необходимы функционал. Обратите, внимание на то, чтобы Ваша реализация методов setTime(...) предусматривала возможность конвертации созданных повторяющихся задачи в неповторяющиеся и наоборот.

Предусмотрите валидацию аргументов методов и конструкторов класса Task с помощью оповещения пользователя (`System.out.println(…)`) о заданных невалидных аргументах методов или конструкторов.

Используйте принцип инкапсуляции ООП при проектировании класса Task.

Тестирование

Добавить в проект тесты, которые находятся в архиве `unit.zip` (будут доступны на занятии), изменив имена пакетов теста в соответствии с правилами, которые даны в начале задания. Обратите внимание что 100% прохождение теста, еще не показатель, что код работает правильно. Сравните желаемое и реально поведение кода.