# Задача 2. Интерпретатор “ЛогоМир”

### Общая информация

Требуется реализовать консольный интерпретатор команд для Абстрактного Исполнителя (АИ) на прямоугольном поле.

### Устройство интерпретатора

1. Команды:
    a. `INIT <width> <height> <x> <y>`
    инициализирует игровое поле заданных размеров, помещая АИ в указанную позиицю на поле.
    Любая “программа” должна начинаться с этой команды (при этом, не запрещается неоднократное использование команды в программе).
    Параметры команды должны осмысленно валидироваться.
    b. `MOVE [L|R|U|D] <steps>` - перемещат АИ по направлению влево|вправо|вверх|вниз (должно быть задано ровно одно направление из `L, R, U, D`) на заданное количество шагов. Поле считается торообразным, т.е. при переходе через границу, движение “закольцовывается”
    c. `DRAW` - АИ переходит в состояние рисования. В этом состоянии, при перемещении по полю командой `MOVE`, АИ оставляет след.
    d. `WARD` - АИ выходит из состояния рисования.
    e. `TELEPORT <x> <y>` - АИ перемещается в указанную координату (не оставляя следов).
2. После старта программы и получения команды `INIT`, программа рисует в консоли заданное поле, любым красивым спобосом, на поле изображается АИ. При получении дальнейших команд, действия АИ или изменения поля соответствующим образом интерпретируются.
3. Необязательное интересное расширение: позволить в программе ветвления и циклы, для этого добавить команду условного перехода `IF_JMP` и считать, что все команды нумеруются. В аргументах `IF_JMP` можно позволить использовать текущие координаты АИ, его состояние, размеры поля - они могут обозначаться какими-то ключевыми словами.

### Технические требования

1. При сдаче продемонстрировать исполнение 2-3 нетривиальных программ.
2. Для создания команд использовать фабрику объектов.
3. Фабрика объектов использует механизм Java Reflection для создания объектов. Фабрика конфигурируется в момент своего создания файлом, в котором указано соответствие между именами команд и полными квалифицированными именами классов, которые соответствуют каждой команде (например,

TELEPORT=ru.nsu.ivanov.LogoWorld.commands.Teleport). После этого, при получении первого запроса на создание объекта по имени TELEPORT, фабрика пытается загрузить указанный класс при помощи метода Class.forName() и в случае успешной загрузки, инстанцирует экземпляр этого класса при помощи Class.newInstance().

Класс каждой команды должен не должен загружаться более одного раза, т.е. загруженные экземпляры java.lang.Class требуется кэшировать.
Фабрика не должна зависеть от конкретных классов команд.

Конфигурационный файл для фабрики загружать с помощью ClassLoader.getResourceAsStream(). Для разбора файла удобно использовать класс java.util.Properties.

4. Классы программы должны быть протестированы с помощью JUnit (либо другого удобного инструмента).
5. Ход выполнения программы должен быть журналирован с помощью библиотеки log4j (либо другой удобной).
6. Основные классы и методы должны быть тщательно документированы в формате javadoc.
7. Сборка, запаковывание в jar-файл и запуск должны быть реализованы с помощью технологии Apache Ant.