TDD шахматы

Для разработки шахматной программы в парадигме Test Driven Development требуется подготовить набор тестов для классов шахматных фигур. Каждая фигура унаследована от класса, имплементирующего интерфейс вида (псевдокод!):

```
interface iChessman {
        TChessField getPosition();
        TChessMove goToPosition(TChessField);
}

class TChessman implements iChessman{
        TChessman(EChessmanType, TChessField, ESide);
}
```

EChessmanType - множество типов шахматных фигур, *ESide* - сторона: белые или черные;

Для удобства работы с шахматной доской введен класс *TChessField*, а сама доска - это массив из 64 константных объектов этого класса (для постановки задачи описание класса самой шахматной доски не принципиально):

```
class TChessField {
        TChessField(char row, char col); // конструктор: принимает позицию
        char getRow();
        char getCol();
        TChessman* isBusy(); // проверяем, стоит ли кто-то на этом поле?
}
```

Задание:

1. На любом (по выбору студента) ООП языке программирования написать тесты для двух любых шахматных фигур по выбору
2. В тестах учесть как негативные (например, попытка хода “за доску” или на недосягаемое поле), так и позитивные варианты (ход на досягаемое поле, взятие другой фигуры)
3. Для выбранной фигуры учесть все возможные игровые ситуации (например, “взятие на проходе” или “превращение” для пешки, рокировку для короля и ладьи)
4. Отдельно протестировать ситуацию взятия одной фигурой другой; для этого разработать не только тест, но и класс *TChessMove* (игровой ход), одним из методов которого должен быть asString(), возвращающий описание хода в виде строки в шахматной нотации. Для класса *TChessMove* и его методов тесты писать не обязательно.
5. Реализовать минимально функциональные классы выбранных фигур и продемонстрировать выполнение тестов.
6. Оформить отчет, в котором описать логику построения ваших тестов и итоговые результаты выполнения. Код приложить отдельно

Примечание: допускается использовать как фреймворки, так и самописные механизмы тестирования, опирающиеся на базовые возможности выбранного языка.