Урок Спринт 1.14

# Пакет world

Теперь нам нужно создать папку `life` в папке `pkg`, и файл `world.go` внутри:

```
package life

type World struct {
    Height int // Высота сетки
    Width  int // Ширина сетки
    Cells  [][]bool
}
// Используйте код из предыдущего урока по игре «Жизнь»
func NewWorld(height, width int) *World {
}
func (w *World) next(x, y int) bool {
}
func (w *World) neighbors(x, y int) int{
}
func NextState(oldWorld, newWorld World)  {
}
// RandInit заполняет поля на указанное число процентов
func (w *World) RandInit(percentage int) {
    // Количество живых клеток
    numAlive := percentage * w.Height * w.Width / 100
    // Заполним живыми первые клетки
    w.fillAlive(numAlive)
    // Получаем рандомные числа
    r := rand.New(rand.NewSource(time.Now().Unix()))

    // Рандомно меняем местами
    for i := 0; i < w.Height*w.Width; i++ {
        randRowLeft := r.Intn(w.Width)
        randColLeft := r.Intn(w.Height)
        randRowRight := r.Intn(w.Width)
        randColRight := r.Intn(w.Height)

        w.Cells[randRowLeft][randColLeft] = w.Cells[randRowRight][randColRight]
    }
}

func (w *World) fillAlive(num int) {
    aliveCount := 0
    for j, row := range w.Cells {
        for k := range row {
            w.Cells[j][k] = true
            aliveCount++
            if aliveCount == num {

                return
            }
        }
    }
}
```

Наш первый пакет почти готов. Осталось самое главное — написать тесты. На это обычно тратится больше времени, чем на сам функционал пакета. Тестировать мы будем экспортируемые функции.

Создайте файл `life_test.go` и внутри объявите пакет `life_test`:

```
package life_test
```

```
// Так как это другой пакет, нужно его импортировать
import "github.com/aivanov/game/pkg/life"
```

Для функции `NewWorld` ожидаемый результат — это выделить память под сетку. Это мы и будем проверять в тесте `TestNewWorld`. Названия функций в тестах обычно состоят из слова `Test` и имени тестируемой функции.

```
func TestNewWorld(t *testing.T) {
    // Задаём размеры сетки
    height := 10
    width := 4
    // Вызываем тестируемую функцию
    world := life.NewWorld(height, width)
    // Проверяем, что в объекте указана верная высота сетки
    if world.Height != height {
        t.Errorf("expected height: %d, actual height: %d", height, world.Height)
    }
    // Проверяем, что в объекте указана верная ширина сетки
    if world.Width != width {
        t.Errorf("expected width: %d, actual width: %d", width, world.Width)
    }
    // Проверяем, что у реальной сетки — заданная высота
    if len(world.Cells) != height {
        t.Errorf("expected height: %d, actual number of rows: %d", height, len(world.Cells))
    }
    // Проверяем, что у каждого элемента — заданная длина
    for i, row := range world.Cells {
        if len(row) != width {
            t.Errorf("expected width: %d, actual row's %d len: %d", width, i, world.Width)
        }
    }
}
```

В реальных тестах стоит проверять вызов функций с проблемными значениями. Например, в нашей программе вызов функции `NewWorld` с отрицательными размерами вызовет панику, а это недопустимо. В таком случае наша функция должна вернуть ошибку. Попробуйте настроить проверку этой функции самостоятельно.

Таким же образом нужно протестировать все оставшиеся функции пакета.