Языки программирования (осень 2018)

Лабораторная работа №6

Моя работа

Инструкции для работы

Для того, чтобы отправить работу на оценку, нажмите "Начало подготовки Вашей работы".

На открывшейся странице:

- в поле **Название** появившегося окна укажите точное название загружаемого файла (строчными буквами, с расширением, без пробелов);
- поле **Содержимое работы** оставьте пустым;
- из папки с решением перетащите загружаемый файл в поле **Приложение** или загрузите файл в это поле, используя кнопку "Добавить.." в меню этого поля;
- выполнив перечисленные пункты нажмите кнопку "Сохранить".

При необходимости, пока не окончена фаза представления работ, можно откорректировать представление работы нажав кнопку "Редактировать работу"

lab-6.rb

представлено: *Среда, 5 Декабрь 2018, 12:49*

- lab-6.rb

Самооценка
от Максим Кулаков
Оценка: 99,24 из 100,00
Вес: 16

Форма оценки

Критерий 1

Метод `add` должен быть определен в рамках класса `Array` и заключаются в получении результата без побочного эффекта. В операции участвуют два массива: первый - `self`, второй - аргумент метода. Метод может быть определен следующим образом

```
class Array
  def add b
    self.zip(b).map {|l| l.reduce(:+)}
  end
end
```

Может быть еще много вариантов определения этого метода. Например,

```
class Array
  def add b
    self.each_with_index.map {|a, i| a + b[i]}
  end
end
```

- Не следует ставить за решение более 1 балла, если для каких-то двух числовых массивов `a` и `b` в результате выполнения

  ```
  c = a.add b
  ```

  изменяется `a` или `b`.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 1

5

Комментарий к Критерий 1

Критерий 2

Определение класса `Field`, включая его инициализацию, может выглядеть следующим образом:

```
class Field
  def initialize
    @field = Array.new(FieldSize) {Array.new(FieldSize)}
  end

  FieldSize = 10

  ...
end
```

(порядок расположения определений метода `initialize` и константы `FieldSize` не имеет значения).

- В методе `initialize` должен корректно определяться двумерный массив. Если это не так, то решение заслуживает не более 1 балла.
  В частности, если второе упоминание `Array.new` является вторым аргументом первого вызова `Array.new`, т.е. имеет место подобный вызов:

  ```
  @field = Array.new(FieldSize, Array.new(FieldSize))
  ```

  то решение заслуживает не более 1 балла.

- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 2

5

Комментарий к Критерий 2

Критерий 3

Метод `size` должен являться методом класса `Field` и возвращать значение константы `FieldSize`. То есть метод должен выглядеть следующим образом

```
def self.size
  FieldSize
end
```

- Решение заслуживает не более 1 балла, если в заголовке метода отсутствует `self.`.
- Решение заслуживает не более 1 балла, если оно сложнее чем то, что представлено выше.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 3

5

Комментарий к Критерий 3

Критерий 4

Метод `set!` может иметь следующий вид.

```
def set! (n, x, y, hor, ship)
  n.times do
    @field[x][y] = ship
    if hor then x += 1 else y += 1 end
  end
end
```

Может быть еще много вариантов определения этого метода. Например,

```
def set! (n, x, y, hor, ship)
  dims = [1, n]
  if hor then dims.reverse! end
  @field[x, dims[0]].each do |row|
    row[y, dims[1]] = Array.new(dims[1], ship)
  end
end
```

- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 4

5

Комментарий к Критерий 4

Критерий 5

Метод `print_field` может иметь следующий вид.

```
def print_field
  print "+"
  (1..Field.size).each {print "-"}
  print "+\n"
  @field.each do |row|
    print "|"
    row.each {|x| print (!x ? " " : x.to_s)}
    print "|\n"
  end
  print "+"
  (1..Field.size).each {print "-"}
  print "+\n"
  nil
end
```

- Следует снизить оценку на 2 балла, если вместо константы `FieldSize` или метода `Field.size` используется число 10 (или другое число, заданное литералом).
- Следует снизить оценку на 1 балл, если вместо методов-итераторов используется оператор цикла `while`.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 5

5

Комментарий к Критерий 5

Критерий 6

Метод `free_space?` может иметь следующий вид.

```
def free_space? (n, x, y, hor)
  field_b = [x,y]
  dims = [0,n-1]
  if hor then dims.reverse! end
  field_e = field_b.add dims
  if (field_b + field_e).all? {|a| a.between?(0, FieldSize - 1)}
    dims = dims.add [1, 1]
    field_b.each_index do |i|
      if field_b[i] > 0
        field_b[i] -= 1
        dims[i] += 1
      end
    end
    field_e.each_index do |i|
      if field_e[i] < FieldSize - 1 then dims[i] += 1 end
    end
    @field[field_b[0], dims[0]].all? do |row|
      row[field_b[1], dims[1]].all? {|cell| !cell || cell == ship}
    end
  else
    false
  end
end
```

- Следует оценить решение не более чем в 1 балл, если в методе не проверяется, что заданные координаты `x` и `y` находятся в диапазоне координат клеток игрового поля.
- Следует оценить решение не более чем в 1 балл, если в методе не проверяется, что заданные параметры определяют корабль, полностью помещающийся в игровое поле.
- Следует снизить оценку на 2 балла, если вместо константы `FieldSize` или метода `Field.size` используется число 10 (или другое число, заданное литералом).
- Следует снизить оценку на 1 балл, если для какой-то клетки (для каких-то клеток) игрового поля дважды осуществляется проверка на то, что она свободна.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 6

5

Комментарий к Критерий 6

Критерий 7

Определение класса `Ship`, включая его инициализацию, может выглядеть следующим образом:

```
class Ship
  attr_reader :coord, :len

  def initialize(field, len)
    @len = len
    @myfield = field
    @maxhealth = 100 * len
    @minhealth = 30 * len
    @health = @maxhealth
  end

  ...
end
```

(порядок расположения определений метода `initialize` и геттеров `coord` и `len` не имеет значения).

- Трудно представить себе решение сложнее, но если такое найдется, следует снизить оценку на 1 балл.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 7

5

Комментарий к Критерий 7

Критерий 8

Метод `to_s` в классе `Ship` должен иметь следующий вид.

```
def to_s
  "X"
end
```

- Следует оценить решение не более чем в 1 балл, если запуск метода возвращает что-то отличное от строки `"X"`.
- Следует оценить решение не более чем в 1 балл, если запуск метода выдает строку `"X"` на экран, а в качестве результата выдает `nil` (это случается, когда в теле метода присутствует вызов процедуры вывода на экран `puts` или `print`).
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 8

5

Комментарий к Критерий 8

Критерий 9

Метод `clear` должен иметь следующий вид.

```
def clear
  @myfield.set!(@len, @coord[0], @coord[1], @hor)
end
```

- Следует оценить решение в 1 балл, если в решении отсутствует вызов `@myfield.set!`.
- Не следует снижать оценку, если в качестве пятого параметра явно передается значение `nil`
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 9

5

Комментарий к Критерий 9

Критерий 10

Метод `set!` в классе `Ship` может иметь следующий вид.

```
def set! (x, y, hor)
  if @myfield.free_space?(@len, x, y, hor, self)
    if @coord then clear end
    @myfield.set!(@len, x, y, hor, self)
    dim = [0, @len - 1]
    if hor then dim.reverse! end
    @coord = [x, y] + ([x, y].add dim)
    @hor = hor
    true
  else
    false
  end
end
```

- Следует оценить решение в 1 балл, если в решении отсутствует любой из вызовов: `@myfield.set!`, `@myfield.free_space?`.
- Следует оценить решение в 1 балл, если в решении в вызове `@myfield.free_space?` последним аргументом передается что-то отличное от `self`
- Следует оценить решение в 1 балл, если в решении в вызове `@myfield.set!` последним аргументом передается что-то отличное от `self`
- Следует оценить решение в 1 балл, если в решении нет ни вызова `clear`, ни вызова `kill` (снижать не нужно, если присутствует вызов какого-то из двух методов).
- Следует оценить решение в 1 балл, если в результате метода в последних двух элементах массива `@coord` сохраняется нечто отличное от координат последней клетки корабля.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 10

5

Комментарий к Критерий 10

Критерий 11

Метод `kill` должен иметь следующий вид.

```
def kill
  clear
  @coord = nil
end
```

- Следует снизить оценку на 2 балла, если в решении отсутствует вызов `clear`.
- Следует оценить решение не более чем в 1 балл, если переменной `@coord` присваивается значение, отличное от `nil`.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 11

5

Комментарий к Критерий 11

Критерий 12

Метод `explode` должен иметь следующий вид.

```
def explode
  @health -= 70
  if @health <= @minhealth then
    kill
    return @len
  end
  nil
end
```

- Следует снизить оценку на 2 балла, если в решении отсутствует вызов `kill`.
- Следует оценить решение не более чем в 1 балл, если при равенстве `@health` и `@minhealth` не происходит вызова `kill` (т.е. случай, эквивалентный приведенному решению, в котором нестрогое неравенство заменено на строгое).
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 12

5

Комментарий к Критерий 12

Критерий 13

Метод `cure` может иметь следующий вид.

```
def cure
  @health += 30
  if @health > @maxhealth then @health = @maxhealth end
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 13

5

Комментарий к Критерий 13

Критерий 14

Метод `health` может иметь следующий вид.

```
def health
  (100 * @health.to_f / @maxhealth).round(2)
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Следует оценить решение в 1 балл, если перед делением ни к делимому, ни к делителю не применяется метод `to_f` .
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 14

5

Комментарий к Критерий 14

Критерий 15

Метод `move` класса `Ship` может иметь следующий вид.

```
def move forward
  moves = [0, forward ? 1 : -1]
  if @hor then moves.reverse! end
  new_coord = @coord.add (moves + moves)
  set!(new_coord[0], new_coord[1], @hor)
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 15

5

Комментарий к Критерий 15

Критерий 16

Метод `rotate` может иметь следующий вид.

```
def rotate (n, k)
  if n.between?(1, @len)
    new_hor = (k % 2 == 1) ? !@hor : @hor
    if k==1
      if @hor
        new_coord = [@coord[0] + n - 1, @coord[1] - n + 1]
      else
        new_coord = [@coord[2] + n - @len, @coord[3] + n - @len]
      end
      set!(new_coord[0], new_coord[1], new_hor)
    elsif k==2
      if @hor
        new_coord = [@coord[2] + 2 * n - 2* @len, @coord[3]]
      else
        new_coord = [@coord[2], @coord[3] + 2 * n - 2* @len]
      end
      set!(new_coord[0], new_coord[1], new_hor)
    elsif k==3
      if @hor
        new_coord = [@coord[2] + n - @len, @coord[3] + n - @len]
      else
        new_coord = [@coord[0] - n + 1, @coord[1] + n - 1]
      end
      set!(new_coord[0], new_coord[1], new_hor)
    else
      false
    end
  else
    false
  end
end
```

- Следует оценить решение не более чем в 1 балл, если в методе не проверяется, что заданные параметры `n` и `k` находятся в в необходимых диапазонах.
- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 16

5

Комментарий к Критерий 16

Критерий 17

Определение класса `BattleField`, включая его инициализацию, может выглядеть следующим образом:

```
class BattleField < Field
  def initialize
    super
    newships
  end

  Ships = [4,3,3,2,2,2,1,1,1,1]

  def newships
    @allships = Ships.map {|len| Ship.new(self, len)}
  end

  ...
end
```

(порядок расположения определений метода `initialize`, константы `Ships` и метода `newships` не имеет значения).

- Следует снизить оценку за решение на один балл, если в методе `initialize` нет вызова `super`.
- Следует оценить решение не более чем на 1 балл, если в методе `newships` вызову `Ship.new` в качестве первого аргумента вместо `self` передается что-то иное.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 17

5

Комментарий к Критерий 17

Критерий 18

Метод `fleet` может иметь следующий вид.

```
def fleet
  @allships.each_with_index.map {|x, i| [i, x.len]}
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Следует оценить решение в 1 балл, если в решении не используется метод `len` объектов класса `Ship`.
- Следует оценить решение в 1 балл, если в массив-результат вносятся в качестве первых элементов пар не номера элементов массива `@allships` а что-то иное.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 18

5

Комментарий к Критерий 18

Критерий 19

Метод `place_fleet` может иметь следующий вид.

```
def place_fleet pos_list
  res = pos_list.inject(true) do |a, l|
    a && @allships[l[0]].set!(l[1], l[2], l[3])
  end
  if res
    res = @allships.inject(true) {|a, ship| a && ship.coord}
  end
  if !res
    @allships.each {|ship| if ship.coord then ship.kill end}
  end
  res
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Следует оценить решение в 1 балл, если в решении не используется метод `set!` объектов класса `Ship`.
- Следует оценить решение в 1 балл, если в решении не используется метод `kill` объектов класса `Ship`.
- Следует оценить решение в 1 балл, если в решении не используется метод `coord` объектов класса `Ship`.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 19

5

Комментарий к Критерий 19

Критерий 20

Метод `remains` может иметь следующий вид.

```
def remains
  @allships.each_with_index.map {|x, i| [i, x.coord, x.len, x.health]}
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Следует оценить решение в 1 балл, если в решении не используется хотя бы один из методов объектов класса `Ship`: `len`, `coord`, `health`.
- Следует оценить решение в 1 балл, если в массив-результат вносятся в качестве первых элементов четверок не номера элементов массива `@allships` а что-то иное.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень

совпадения решения с приведенными вариантами.

Оценка для Критерий 20

5

Комментарий к Критерий 20

Критерий 21

Метод `refresh` может иметь следующий вид.

```
def refresh
  @allships = @field.reduce(:|).find_all {|x| x}
end
```

Вопрос по заданию стоит: оставить в массиве только те корабли, которые размещены в игровом поле. Поэтому обновление массива должно исходить из содержимого игрововго поля, а не из исходного содержимого массива.

- Следует снизить оценку на 1 балл, если обновление массива происходит через анализ элементов массива, например:

  ```
  def refresh
    @allships = @allships.find_all {|x| x.coord}
  end
  ```

- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 21

5

Комментарий к Критерий 21

Критерий 22

Метод `shoot` может иметь следующий вид.

```
def shoot c
  if @field[c[0]][c[1]]
    if res = @field[c[0]][c[1]].explode
      refresh
      "killed #{res}"
    else
      "wounded"
    end
  else
    "miss"
  end
end
```

- Следует снизить оценку на 2 балла, если в методе не происходит прямого обращения к массиву `@field`.
- Следует оценить решение в 1 балл, если в решении не используется хотя бы один из методов: `explode`, `refresh`.
- Следует оценить решение в 1 балл, если в решении происходит прямое обращение к массиву `@allships`.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 22

5

Комментарий к Критерий 22

Критерий 23

Метод `cure` должен иметь следующий вид.

```
def cure
  @allships.each {|ship| ship.cure}
end
```

- Следует снизить оценку на 2 балла, если при правильной работе метода нет обращения к методу `cure` объекта класса `Ship`.
- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 23

5

Комментарий к Критерий 23

Критерий 24

Метод `game_over?` должен иметь следующий вид.

```
def game_over?
  @allships.empty?
end
```

- Следует оценить решение не более чем на 1 балл, если вместо проверки пустоты массива `@allships`, т.е. наличия в нем 0 элементов, или сравнения его с пустым массивом `[]`, массив сравнивается со значением `nil`.
- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 24

5

Комментарий к Критерий 24

Критерий 25

Метод `move` объектов класса `BattleField` может иметь следующий вид.

```
def move l_move
  if l_move[1].between?(1,3)
    @allships[l_move[0]].rotate(l_move[2], l_move[1])
  else
    @allships[l_move[0]].move(l_move[2] == 1)
  end
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 25

5

Комментарий к Критерий 25

Критерий 26

Определение класса `Player`, включая его инициализацию, может выглядеть следующим образом:

```
class Player
  attr_reader :name
  attr_accessor :manual
  def initialize (name, manual = true)
    @name = name
    @manual = manual
    @lastsample = [1, 0]
    reset
  end

  def reset
    @lastshots = []
    @allshots = []
  end

  ...
end
```

(порядок расположения определений метода `initialize`, `name`, `manual` и `reset` не имеет значения).

- Следует снизить оценку на 2 балла, если у последнего параметра метода `initialize` не указано значение по умолчанию или значение по умолчанию отлично от заданного.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 26

5

Комментарий к Критерий 26

Критерий 27

Метод `random_point` может иметь следующий вид.

```
def random_point
  [rand(Field.size), rand(Field.size)]
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Следует снизить оценку на 2 балла, если вместо значения метода `Field.size` используется число 10 (или другое число, заданное литералом).
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 27

5

Комментарий к Критерий 27

Критерий 28

Метод `place_strategy` может иметь следующий вид.

```
def place_strategy ship_list
  tmp_field = Field.new
  dirs = [true, false]
  res = []
  (ship_list.sort {|x,y| y[1] <=> x[1]}).each do |s|
    flag = false
    while !flag
      p = random_point
      hor = dirs.sample
      if tmp_field.free_space?(s[1], p[0], p[1], hor, s[0])
        tmp_field.set!(s[1], p[0], p[1], hor, s[0])
        res.push [s[0], p[0], p[1], hor]
        flag = true
      end
    end
  end
  res
end
```

- Следует снизить оценку на 2 балла, если вместо значения метода `Field.size` используется число 10 (или другое число, заданное литералом).
- Следует снизить оценку на 1 балл, если не используется метод `random_point`.
- Следует снизить оценку на 2 балла, если при выборе разворота корабля вместо случайного разворота отдается предпочтение одному из направлений (т.е. присутствует проверка: если можно поставить корабль в данном направлении, то поставить именно так, а иначе - только при невозможности первого).
- Следует оценить решение в 1 балл, если расстановка кораблей ведется не в порядке убывания их длины. В связи с этим, обратите внимание, что метод `sort`, в отличие от `sort!`, не изменяет исходный массив, а выдает новый.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 28

5

Комментарий к Критерий 28

Критерий 29

Методы `hit` и `miss` могут иметь следующий вид.

```
def hit message
  @lastshots.push [@shot, message]
end
def miss
  @allshots.push( hit("miss") )
  @lastshots = []
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Не следует снижать оценку, если в `miss` не используется `hit`.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 29

5

Комментарий к Критерий 29

Критерий 30

Необходимый фрагмент метода `shot_strategy` может иметь следующий вид.

```
if @lastshots.length == 0 || @lastshots[-1][1][0,6] == "killed"
  @shot = random_point
else
  if @lastshots.length == 1 || @lastshots[-2][1][0,6] == "killed"
    @lastsample = [[0,1],[0,-1],[1,0],[-1,0]].sample
    @shot = @lastshots[-1][0]
  end
  @shot = @shot.add @lastsample
  if ! @shot.all? {|x| x.between?(0, Field.size-1)}
    @lastsample = @lastsample.map {|x| -x}
    @shot = (@lastshots[-1][0]).add @lastsample
  end
end
if @lastshots.any? {|x| x[0] == @shot}
  shot_strategy
else
  @shot
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Следует снизить оценку на 2 балла, если вместо значения метода `Field.size` используется число 10 (или другое число, заданное литералом).
- Следует снизить оценку на 1 балл, если не используется метод `random_point`.
- Следует снизить оценку на 2 балла, если при втором выстреле по одному кораблю вместо случайной соседней точки отдается предпочтение одному из направлений (т.е., например, только вправо от предыдущего выстрела).
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень

совпадения решения с приведенными вариантами.

Оценка для Критерий 30

5

Комментарий к Критерий 30

Критерий 31

Необходимый фрагмент метода `ship_move_strategy` может иметь следующий вид.

```
weakest = (remains.sort {|a, b| a[3] <=> b[3]})[0]
[weakest[0], rand(4), rand(1..weakest[2])]
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 31

4

Комментарий к Критерий 31

Критерий 32

Определение класса `Game`, включая его инициализацию, может выглядеть следующим образом:

```
class Game
  def initialize (player_1, player_2)
    @players= [[player_1, BattleField.new, 0],
               [player_2, BattleField.new, 0]]
    @players.each {|p| reset p}
    @players.shuffle!
    @game_over = false
  end

  def reset p
    print(p[0].name, " game setup\n")
    p[0].reset
    player_ships = p[1].fleet
    if !p[1].place_fleet(p[0].place_strategy player_ships)
      raise "Illegal ship placement"
    else
      puts "Ships placed"
    end
  end

  ...
end
```

(порядок расположения определений методов `initialize` и `reset` не имеет значения).

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Следует оценить решение в 1 балл, если в `initialize` имеется не точно два вызова `BattleField.new`.
- Следует оценить решение в 1 балл, если первый ход делает всегда первый игрок, т.е. список игроков не перемешивается после инициализации. В связи с этим, обратите внимание, что метод `shuffle`, в отличие от `shuffle!`, не изменяет исходный массив, а выдает новый.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 32

5

Комментарий к Критерий 32

Критерий 33

Метод `start` может иметь следующий вид.

```
def start
  lastshots = []
  while ! @game_over
    p1 = @players[0]
    p2 = @players[1]
    p1[2] += 1
    print("Step #{p1[2]} of player ",p1[0].name, "\n")
    p1[1].cure
    p1[1].move (p1[0].ship_move_strategy p1[1].remains)
    shot = p1[0].shot_strategy
    if lastshots.include? shot
      puts "Illegal shot"
      res = "miss"
    else
      lastshots.push shot
      res = p2[1].shoot shot
    end
    print(shot, " ", res, "\n")
    if res == "miss"
      p1[0].miss
      @players.reverse!
      lastshots = []
    else
      p1[0].hit res
      @game_over = p2[1].game_over?
      if @game_over
        puts "Player #{p1[0].name} wins!"
        break
      end
    end
  end
end
```

- Следует снизить оценку на 1 балл, если решение сложнее, чем приведено в примере.
- Ваша оценка должна быть беспристрастной. Вполне нормально, если решение отличается от приведенных выше вариантов. Вы проверяете правильность решения и его стиль, а не степень совпадения решения с приведенными вариантами.

Оценка для Критерий 33

5

Комментарий к Критерий 33

- ЯП 2018
  - Участники
  - Значки
  - Компетенции
  - Оценки
  - Общее
  - Форумы курса
  - Материалы по тематике курса
  - Лабораторные работы
  - Оценка лабораторных работ
  - Лабораторная работа №0
  - Лабораторная работа №1
  - Лабораторная работа №2
  - Лабораторная работа №3
  - Лабораторная работа №4
  - Лабораторная работа №5
  - **Лабораторная работа №6**
    - Моя работа
  - Раздел 1. Standard ML
  - Раздел 2. Haskell
  - Раздел 3. LISP
  - Раздел 4. Ruby
  - Раздел 5. PROLOG
  - Реферат
  - Аттестация
  - Бонусы
  - Напоминалки
- ИКБ
- On-line
