**Proyecto POOB**

**Jeisson Steban Casallas Rozo y Camilo Nicolas Murcia Espinosa**

**Ciclo 1**

**Iceepeecee:**

* Es el constructor de la clase Iceepeecee, en donde inicializamos el tablero dándole un largo y ancho como mapa que vamos a utilizar, haciendo un llamado a la clase polygonICPC, la cual llama al Canvas respectivamente.

**No lleva pruebas de unidad.**

**addIsland:**

* Método que crea una isla con forma de poligono con un color dado y unas coordenadas ingresadas en forma de matriz, la cual según su longitud indican la cantidad de vértices que va a tener la isla y la posición de esta.

**Diagrama de secuencia:**

**Pruebas de unidad:**

**AccordingCMShould:**

* **Se crea una isla correctamente:** Nos permite crear una isla con un color dado y unas coordenadas dadas de forma correcta.

**AccordingCMShouldNot:**

* **No se pueden crear islas del mismo color**
* **Una isla tiene mínimo 3 vértices**

**delIsland:**

* Metodo que nos permite eliminar una isla que se encuentre en nuestro tablero Iceepecee.

**Pruebas de unidad:**

**AccordingCMShould**

* **Se borra una isla existente**

**AccordingCMShouldNot:**

* **No puede borrar una isla que no existe**

**islandLocation:**

* Este método nos devuelve una matriz con las coordenadas de la isla dados los vértices de la misma.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Localización de una isla existente:**

**AccordingCMShouldNot:**

* **No puede obtener la localización de una isla no existente:**

**addFlight:**

* Nos permite crear un vuelo el cual es visto como una recta con un color dado por el usuario y dos coordenadas en forma de arreglo, una donde inicia y otra donde termina.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Se crea un vuelo correctamente**

**AccordingCMShouldNot:**

* **No se pueden crear islas del mismo color:**
* **No puede recibir menos de 3 coordenadas**

**delFlight:**

* Metodo que nos permite eliminar una isla que se encuentre en nuestro tablero Iceepecee.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Se borra un vuelo existente:**

**AccordingCMShouldNot:**

* **No puede borrar un vuelo que no existe**

**flightLocation:**

* Este método nos devuelve una matriz con las coordenadas del vuelo dadas segun los vértices de inicio y final de la misma.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Localización de un vuelo existente:**

**AccordingCMShouldNot:**

* **No puede obtener la localización de un vuelo no existente:**

**photograph:**

* Con este método el avión especificado por el usuario que realiza la trayectoria toma una foto la cual se ve reflejada del mismo color que el vuelo y la cual tiene un ángulo de apertura dado por el usuario.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Toma una foto del vuelo especificado correctamente**
* **Puede tomar una foto al mismo vuelo con un angulo diferente**

**AccordingCMShouldNot:**

* **No se toma foto a un vuelo que no existe**
* **No puede tomar angulos negativos**
* **No puede tomar angulos mayores a 90**

**photograph2:**

* Con este método todos los aviones que realizan una trayectoria toman una foto la cual se ve reflejada del mismo color que el vuelo y la cual tiene un ángulo de apertura dado por el usuario.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Toma una foto de todos los vuelos correctamente**
* **El angulo de apertura de la foto cambia para todos los vuelos**

**AccordingCMShouldNot:**

* **No se toma foto si no existen vuelos**
* **No puede tomar angulos negativos**
* **No puede tomar angulos mayores a 90**

**flightCamera:**

* Obtenemos el ángulo de la foto especificada por el usuario según su color.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Retorna el angulo de una foto existente correctamente**

**AccordingCMShouldNot:**

* **No puede retornar el angulo de una foto inexistente.**

**makeInvisible:**

* Nos hace Invisibles todos los objetos que se encuentren creados sobre el tablero de iceepeecee

**No llevan pruebas de unidad.**

**makeVisible:**

* Nos hace Visibles todos los objetos que se encuentren creados sobre el tablero de iceepeecee.

**No llevan pruebas de unidad.**

**finish:**

* Termina la simulación del Iceepecee

**No llevan pruebas de unidad.**

**ok:**

* Verifica que el ultimo método ejecutado se haya podido realizar correctamente.

**No llevan pruebas de unidad.**

**RETROSPECTIVA**

**1. ¿Cuáles fueron los mini-ciclos definidos? Justifíquenlos.**

* En cuanto a mini-Ciclos, tenemos todo el ciclo 1 definido entre sus clases, que son islads, flight y photograph, teniendo en cuenta que cada uno tiene sus métodos definidos.

**2. ¿Cuál es el estado actual del proyecto en términos de mini-ciclos? ¿por qué?**

* El ciclo 1 esta completo, su astah terminado, su codigo documentado y probando, junto con pruebas de unidad que lo verifican.

**3. ¿Cuál fue el tiempo total invertido por cada uno de ustedes? (Horas/Hombre)**

* 40 horas por persona aproximadamente.

**4. ¿Cuál consideran fue el mayor logro? ¿Por qué?**

* Lograr crear el simulador con las especificaciones dadas, debido a la cantidad de horas que se han invertido.

**5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?**

* Aprender a definir pruebas relevantes y modelar la situación antes de codificar la solución, este proceso requirió tiempo de planeación de cómo iba a ser la ejecución del programa y los posibles casos en los que podía salir mal.

**6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?**

* + La programación a pares que se vio en las practicas XP, ya que fue clave para poder culminar el ciclo 1, nos comprometemos a seguir así.

**7. Considerando las prácticas XP incluídas en los laboratorios. ¿cuál fue la más útil? ¿por qué?**

* + La práctica más útil fue la del trabajo a pares, pues así se pueden compartir ideas respecto a lo que debemos hacer y enriquecemos el conocimiento mutuo.

**Ciclo 2**

**Iceepeecee:**

* Constructor de la clase Iceepecee que crea el tablero con las islas y los vuelos ingresados

**No lleva pruebas de unidad.**

**Photograph3:**

* Con este método el avión especificado por el usuario que realiza la trayectoria toma una foto la cual se ve reflejada del mismo color que el vuelo y la cual tiene un ángulo de apertura dado por el usuario. A diferencia del primero este recibe ángulos flotantes.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Toma una foto del vuelo especificado correctamente**
* **Puede tomar una foto al mismo vuelo con un angulo diferente**

**AccordingCMShouldNot:**

* **No se toma foto a un vuelo que no existe**
* **No puede tomar angulos negativos**
* **No puede tomar angulos mayores a 90**

**islands:**

* Nos permite consultar las islas identificados con su color que se encuentran en el tablero del iceepeece arrojándonos un arreglo.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Obtiene el arreglo de islas en el tablero.**
* **Obtiene arreglo vacio cuando no hay islas.**

**AccordingCMShouldNot:**

* **Es un método de consulta.**

**flights:**

* Nos permite consultar los vuelos identificados con su color que se encuentran en el tablero del iceepeece arrojándonos un arreglo .

**Pruebas de unidad:**

**AccordingCMShould:**

* **Obtiene el arreglo de vuelos en el tablero.**
* **Obtiene arreglo nulo cuando no hay vuelos.**

**AccordingCMShouldNot:**

* **Es un método de consulta.**

**observedIslands:**

* permite consultar las islas que fueron registradas en una fotografía en su totalidad.

**Pruebas de unidad:**

**AccordingCMShould:**

* **Obtiene las islas cuya fotografía abarca en su totalidad.**
* **Obtiene arreglo vacío cuando ninguna isla es observada en su totalidad o no hay vuelos.**

**AccordingCMShouldNot:**

* **Es un método de consulta.**

**uselessFlights:**

* Permite consultar los vuelos cuya fotografía no es tomada correctamente, es decir, su ángulo de foto no abarca a la isla en su totalidad.

**Pruebas de unidad:**

**AccordingCMShould:**

* **obtiene los vuelos que la foto solo abarco una parte de la isla.**
* **obtiene los vuelos que la foto no abarco ninguna isla.**
* **Obtiene arreglo vacio cuando ninguno es inútil o no hay vuelos.**

**AccordingCMShouldNot:**

* **Es un método de consulta.**

**RETROSPECTIVA**

**1. ¿Cuáles fueron los mini-ciclos definidos? Justifíquenlos.**

* En cuanto a mini-Ciclos, tenemos todo el ciclo 2 como uno solo, separado del ciclo 1, con sus extensiones como lo son métodos de consulta, el otro photograph y su nuevo constructor que facilita las pruebas del simulador.

**2. ¿Cuál es el estado actual del proyecto en términos de mini-ciclos? ¿por qué?**

* Se encuentra realizado en su totalidad, debido a que se le dedicaron las horas necesarias.

**3. ¿Cuál fue el tiempo total invertido por cada uno de ustedes? (Horas/Hombre)**

* Aproximadamente 15 horas por persona.

**4. ¿Cuál consideran fue el mayor logro? ¿Por qué?**

* Implementar los métodos de consultas con respecto a las islas observadas y los vuelos inútiles, debido a la lógica que implicaba la implementación de este.

**5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?**

* Que el código compilara, pero no nos arrojara lo que deseábamos. Se resolvió refactorizando el código.

**6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?**

* + Como equipo hicimos bien el cada uno aportar algo al proyecto para que este saliera de manera más rápida, nos comprometemos a seguir así.

**7. Considerando las prácticas XP incluidas en los laboratorios. ¿cuál fue la más útil? ¿por qué?**

* El trabajo a pares siguió siendo la práctica más útil, donde nos íbamos turnado para realizar diferentes tareas del laboratorio.

**CICLO 3**

**Solve:**

* Método que retorna el ángulo mínimo con el que pueden ser vistas todas las islas del simulador.

**Pruebas de unidad:**

* Se haya el mínimo ángulo correctamente.
* No podría resolver el problema en caso de que el ángulo sea mayor a 90

**Simulate:**

* Método que nos va a simular las islas, con sus vuelos y con sus fotos en el tablero, donde la foto va a tener el ángulo obtenido en el método solve.

**RETROSPECTIVA**

**1. ¿Cuáles fueron los mini-ciclos definidos? Justifíquenlos.**

* Debido a que es otra clase, toda esta hace parte del ciclo 3

**2. ¿Cuál es el estado actual del proyecto en términos de mini-ciclos? ¿por qué?**

* El ciclo 3 se encuentra realizado en su totalidad, sin embargo harian falta probar con mas casos.

**3. ¿Cuál fue el tiempo total invertido por cada uno de ustedes? (Horas/Hombre)**

* Aproximadamente 3 horas por persona.

**4. ¿Cuál consideran fue el mayor logro? ¿Por qué?**

* Implementar la búsqueda binaria para que funcionara el método y relacionarlo con lo que ya teníamos.

**5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?**

* Que el programa se ejecutara sin errores y nos arrojara el angulo indicado. Se resolvió dedicándole tiempo al ciclo.

**6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?**

* + Como equipo hicimos bien el cada uno aportar algo al proyecto para que este saliera de manera más rápida, nos comprometemos a seguir así.

**7. Considerando las prácticas XP incluidas en los laboratorios. ¿cuál fue la más útil? ¿por qué?**

* El trabajo a pares siguió siendo la práctica más útil, donde nos íbamos turnado para realizar diferentes tareas del laboratorio.

**CICLO 4**

**addFlight:** Nos permite agregar un vuelo de un tipo especifico escogido por el usuario, donde cada uno tiene algo que lo hace especial.

**Pruebas de unidad:**

**AccordingCMShould:**

* Se crea una FixedIsland correctamente
* Se crea una SurprisingIsland correctamente
* Se eliminan los puntos de SurprisingIsland Correctamente
* Se mantienen los mismo puntos en SurprisingIsland cuando es igual a 3
* Se crea una ExpandingIsland correctamente.
* La ExpandingIsland crece cada que se llama al makeVisible()

**AccordingCMShouldNot:**

* La fixedIsland no puede ser borrada
* No se pueden crear islas con menos de 3 vertices
* Una isla no puede crear colores repetidos

**addIsland:** Nos permite agregar una isla de un tipo especifico escogido por el usuario, donde cada uno tiene algo que lo hace especial.

**Pruebas de unidad:**

**Pruebas de unidad:**

**AccordingCMShould:**

* Se crea un LazyFlight correctamente
* LazyFlight toma solo una foto
* Se crea un FlatFlight correctamente
* El FlatFlight tiene la misma altura para todo el vuelo.

**AccordingCMShouldNot:**

* Un vuelo no puede tener colores repetidos
* No se pueden ingresar menos de 3 coordenadas en los vuelos

**RETROSPECTIVA**

**1. ¿Cuáles fueron los mini-ciclos definidos? Justifíquenlos.**

* Esta dividido en la implementación de la herencia en island, la implementación de la herencia en flight y la implementación de estos dos métodos en la clase principal Iceepeecee

**2. ¿Cuál es el estado actual del proyecto en términos de mini-ciclos? ¿por qué?**

* El ciclo 4 se encuentra realizado en su totalidad, sin embargo hace falta comprobar su extensibilidad.

**3. ¿Cuál fue el tiempo total invertido por cada uno de ustedes? (Horas/Hombre)**

* Aproximadamente 8 horas por persona

**4. ¿Cuál consideran fue el mayor logro? ¿Por qué?**

* La implementación del fixedIsland pues fue lo que mas problema nos causo al momento de que no quisiéramos que se borrara del diccionario.

**5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?**

* El fixedIsland, debido a que nos obligo a pensar de manera distinta la implementación en varias ocasiones, al final llegamos a la mas optima que se nos ocurrió.

**6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?**

* + El trabajo a par, pues gracias a esto los dos íbamos realizando el ciclo y nos íbamos ayudando en caso de que alguno tuviera dificultad, nos comprometemos a seguir así.

**7. Considerando las prácticas XP incluidas en los laboratorios. ¿cuál fue la más útil? ¿por qué?**

* La simplicidad del código, pues tratamos de que la implementación de la herencia quedara lo mas simple posible.