**Práctica 1 Método Científico**

**Ejercicio 3.**

**Dada la siguiente implementación de una pila.**

**public class Stack {**

**private Integer[] arrayStack;**

**private Integer MAX\_SIZE = 5;**

**private Integer top;**

**/\*\***

**\* Create a stack LIFO with finite deep.**

**\*/**

**public Stack() {**

**this.arrayStack = new Integer[MAX\_SIZE];**

**this.top = 1;**

**}**

**/\*\***

**\* Inserts an element in the stack. If the stack is full, it shows a message**

**\* and stops**

**\***

**\* @param element**

**\* Element to be inserted.**

**\*/**

**public void push(Integer element) {**

**if (top >= MAX\_SIZE) {**

**System.out.println("The stack is full");**

**}**

**arrayStack[this.top] = element;**

**top++;**

**}**

**/\*\***

**\* Extracts an element from the stack. If the stack is empty, it shows a**

**\* message and returns -1.**

**\***

**\* @return Extracted element.**

**\*/**

**public Integer pull() {**

**if (top <= 0) {**

**System.out.println("The stack is empty");**

**return 0;**

**}**

**top--;**

**return arrayStack[top - 1];**

**}**

**/\*\***

**\* Returns the number of elements in the stack.**

**\***

**\* @return Number of elements in the stack.**

**\*/**

**public Integer getSize() {**

**return top;**

**}**

**/\*\***

**\* Extract nElements objects of the stack.**

**\***

**\* @param nElements**

**\* Number of elements to be extracted.**

**\* @return Array with the extracted elements.**

**\*/**

**public Integer[] pull(Integer nElements) {**

**Integer[] res = new Integer[nElements];**

**for (int i = 0; i < nElements; i++) {**

**res[i] = pull();**

**}**

**return res;**

**}**

**/\*\***

**\* Copy an stack into a new stack. The new stack has the same elements in**

**\* the same order as the old one.**

**\***

**\* @return Duplicated stack.**

**\*/**

**public Stack copyStack() {**

**Stack newStack = new Stack();**

**for (int i = 0; i < this.getSize(); i++) {**

**newStack.push(arrayStack[i]);**

**}**

**return newStack;**

**}**

**}**

**Solución**

|  |  |
| --- | --- |
| **Hipótesis** | El programa funciona correctamente. |
| **Predicción** | Al ejecutarlo la salida será correcta. |
| **Experimento** | Ejecutamos con los valores de prueba propuestos en el enunciado. |
| **Observación** | El resultado será correcto solamente para la primera pila. El programa no funciona correctamente. (Al probarlas por separado, en la 2 y 3 aparecerá un error de ejecución y en la 4 contará mal los elementos que hay en ésta). |
| **Conclusión** | Hipótesis rechazada. |

|  |  |
| --- | --- |
| **Hipótesis** | Cambiamos la posición del top del 1 al 0 en el constructor. |
| **Predicción** | Al ejecutar el programa, éste funcionará correctamente. |
| **Experimento** | Ejecutamos con los valores de prueba del enunciado. |
| **Observación** | El programa devuelve un resultado incorrecto (error de ejecución). Depurando el programa nos hemos dado cuenta que el cambio es correcto y el fallo proviene de otro sitio. |
| **Conclusión** | Hipótesis aceptada. |

|  |  |
| --- | --- |
| **Hipótesis** | En el método pull cambiamos la posición del vector (asignamos en top en vez de top-1). |
| **Predicción** | Al ejecutar el código la salida será correcta. |
| **Experimento** | Ejecutamos con los valores de prueba del enunciado. |
| **Observación** | El programa devuelve un resultado correcto para todas las pilas. |
| **Conclusión** | Hipótesis aceptada. Ejercicio resuelto. |