ALGORITMOS Y ESTRUCTURAS DE DATOS – CB100 Curso Schmidt (3° instancia)
26 de Junio de 2025

Padrón: **Apellido y Nombre:**

1. Mostrar la salida y detallar el trabajo del GC

```
public static void main(String[] args) {
    int X, Y, Z = UltimoDigitoPadron;
    char W;
    int T;
    int[] A = new int[4];
    int[] B = A;

    T = 77;
    W = 'C';
    X = T;
    A[0] = X;
    B[1] = A[0] / 2;

    System.out.println(X + "-" + W + "-" + B[0] + "-" + A[1]);

    W = (char)(T + 1);
    B[2] = A[0] * 2;

    System.out.println(W + "-" + T + "-" + B[1] + "-" + A[2]);

    while (A[0] > 30) {
        T -= 2;
        X -= 3;
        Y = X + Z;
        System.out.println(Y + "-" + T + "-" + B[1] + "-" + Z);
        Z = Y;
        A[0] -= 10;
    }

    // ¿Qué hace el garbage collector aquí? ¿Sobre que variable?
}
```

2.
Diseñar la especificación e implementar el TDA Escudo. Debe proveer operaciones para:
• Crearlo recibiendo como parámetro los puntos de resistencia que tiene disponibles.
• Defender Ataque: recibiendo como parámetro la cantidad de puntos del ataque, consume puntos de resistencia.
• Esta Destruido: indica si el Escudo ya agotó sus puntos de resistencia.
• Reparar: restaura la resistencia original, solamente si aún no fue destruido previamente.
• Contar Ataques Soportados: devuelve la cantidad de ataques que pudo defender sin ser destruido.

3.
Implementar el método **elegirPlanMasEconomico** de la clase **AsistenteComercial** a partir de las siguientes especificaciones:

```
class AsistenteComercial {

 public:

  /* post: selecciona de 'planesDisponibles' aquel PlanComercial de menor precio y que tenga los servicios indicados
   *      en 'serviciosRequeridos' como no adicionales.
   */
  PlanComercial elegirPlanMasEconomico(Lista<PlanComercial> planesDisponibles, List<string> serviciosRequeridos);

};
```

```
class PlanComercial {

 public:
  /* post: plan creado con el precion [$] indicado
   */
  PlanComercial(unsigned int precio);

  /* post: devuelve el precio [$] del plan.
   */
  unsigned int obtenerPrecio();

  /* post: devuelve los servicios que brinda el plan.
   */
  Lista<Servicios> obtenerServicios();
};
```

```
class Servicio {

 public:
  /* post: inicializa el Servicio con el nombre indicado.
   */
  Servicio(string nombre, bool adicional);

  /* post: devuelve el nombre que identifica el Servicio.
   */
  string obtenerNombre();

  /* post: indica si el servicio es o no adicional.
   */
  bool esAdicional();
  };
```

Se aprueba por punto, para tener bien el punto hay que tener el 60% bien del ejercicio.
Una vez empezado el examen, no se puede utilizar el celular y no se puede salir del aula.
Duración del examen: 2:00 horas