UNIVERSIDAD NACIONAL DE CAJAMARCA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS

ALGORITMOS Y ESTRUCTURA DE DATOS II
03: OPERACIONES PARA MANIPULAR LISTAS ENLAZADAS SIMPLES

INTRODUCCIÓN

Semana 03 - Operaciones para manipular listas enlazadas simples

PLANTEE UNA SOLUCIÓN

- De los métodos implementados en una lista enlazada:
- ¿Qué método implica mayor trabajo?
- ¿Qué podría a ver para solucionarlo?

RECORDEMOS

- Escriba el código para implementar un Nodo de la lista enlazada
- ¿Tendrías sentido sobrecargar el constructor, implementar los getters and stters, además del método toString?
- ¿Cómo podría saber cuantos elementos tiene una lista?
- Usando un gráfico, indique los pasos a seguir para insertar un elemento al final de la lista

¿CÓMO LO SOLUCIONO?

- Si me piden insertar un elemento al final de la lista de tal modo que no tenga que recorrer toda la lista antes de poder insertarlo

LOGRO ESPERADO

- Al finalizar la sesión, el estudiante programa un menú de opciones para gestionar los datos de una agenda telefónica usando listas enlazadas, implementando las diferentes opciones para cada tarea a realizar sobre la lista enlazadas.

DESARROLLO DEL TEMA

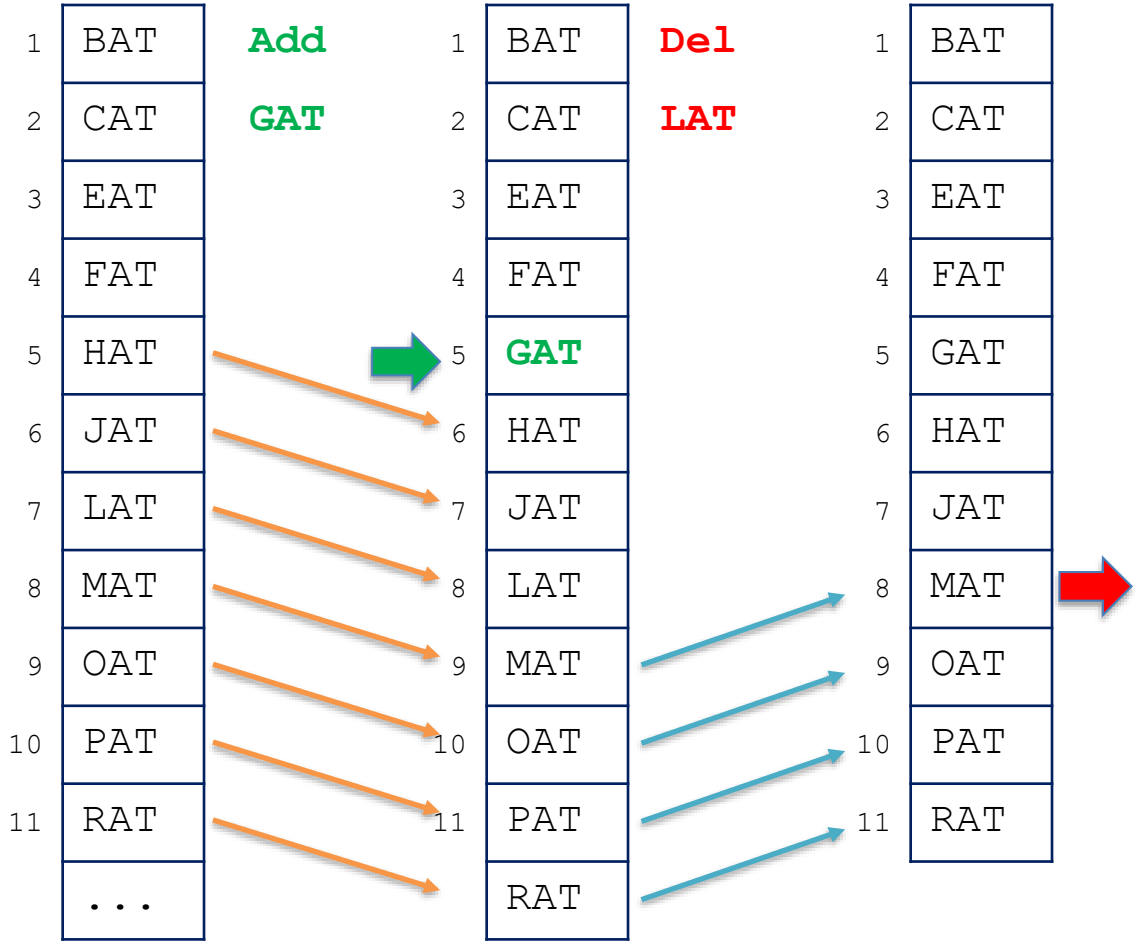
Estructura de Datos: Operaciones para manipular una lista

LISTAS ENLAZADAS REPRESENTACIÓN

- **Representación Vectorial**
- [BAT, CAT, EAT, FAT, HAT, JAT, LAT, MAT, OAT, PAT, RAT, SAT, TAT, VAT,WAT]

- **Representación secuencial (imagen derecha)**

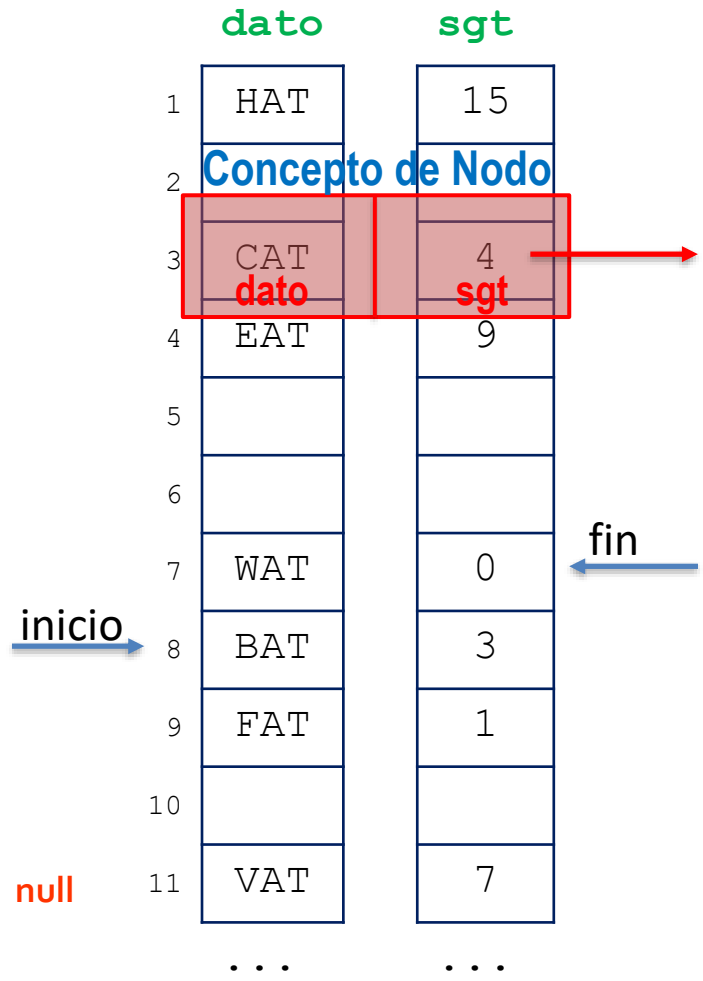
- Los requerimientos de memoria son mínimos.
- Observar que las inserciones y eliminaciones son muy costosas.
- Los nodos están en secuencia en memoria.
- No se requiere una variable de comienzo de lista.



LISTAS ENLAZADAS REPRESENTACIÓN

- **Representación enlazada**
- Los requerimientos de memoria son mayores debido al campo de enlace (sgt).
- Observar que las inserciones y eliminaciones no son muy costosas.
- Los nodos no están en secuencia en memoria.
- Se requiere una variable de comienzo de lista (inicio).

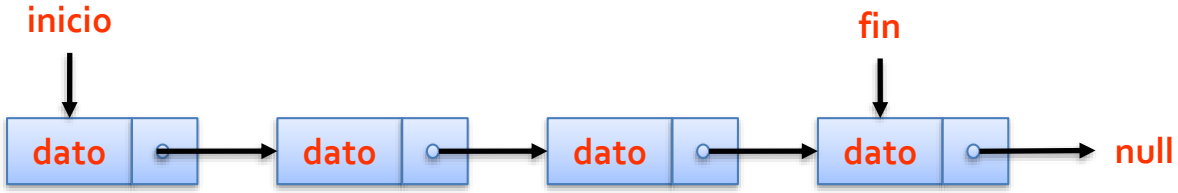
- **Representación gráfica**
- Observar que se trata de una representación ficticia.
- No existe más halla de la representación a fines didácticos.
- Un gráfico vale mil palabras.



LISTAS ENLAZADAS CON DOS REFERENCIAS

- Al estudiar las listas enlazadas se observa que el método que consume mas tiempo es el método para insertar un nodo al final ya que antes de hacerlo tiene que recorrer los n nodos de la lista.
- Una manera de aliviar este problema mejorando la eficiencia del algoritmo es crear una referencia para el último nodo

- De este modo se tiene dos referencias, incio (que ahora sería el nombre de la lista) y fin que referencia al último elemento.



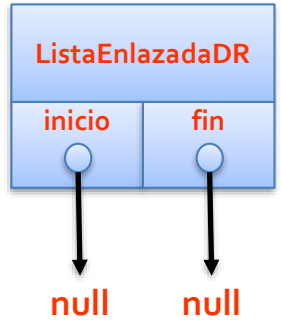
- Esto hará as eficiente el manejo de una cola como se verá mas adelante.
- Además observe qué pasaría si dato deja de ser un entero y se necesita que almacene la información de un Curso (nombre, créditos)

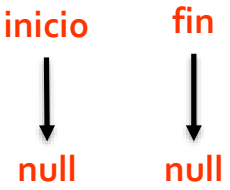
LISTA ENLAZADA DOS REFERENCIAS - DECLARACIONES

```java
public class Curso {
  private String nombre;
  private int creditos;

  public Curso(String nombre, int creditos) {
    super();
    this.nombre = nombre;
    this.creditos = creditos;
  }
  @Override
  public String toString() {
    return nombre + " (" + creditos + ")";
  }
  public String getNombre() {
    return nombre;
  }
  public void setNombre(String nombre) {
    this.nombre = nombre;
  }
  public int getCreditos() {
    return creditos;
  }
  public void setCreditos(int creditos) {
    this.creditos = creditos;
  }
}
```

```java
public class Nodo {
  private Curso dato;
  private Nodo sgt;

  public Nodo(Curso dato, Nodo sgt) {
    super();
    this.dato = dato;
    this.sgt = sgt;
  }

  public Nodo(Curso dato) {
    this(dato, null);
  }
}
```

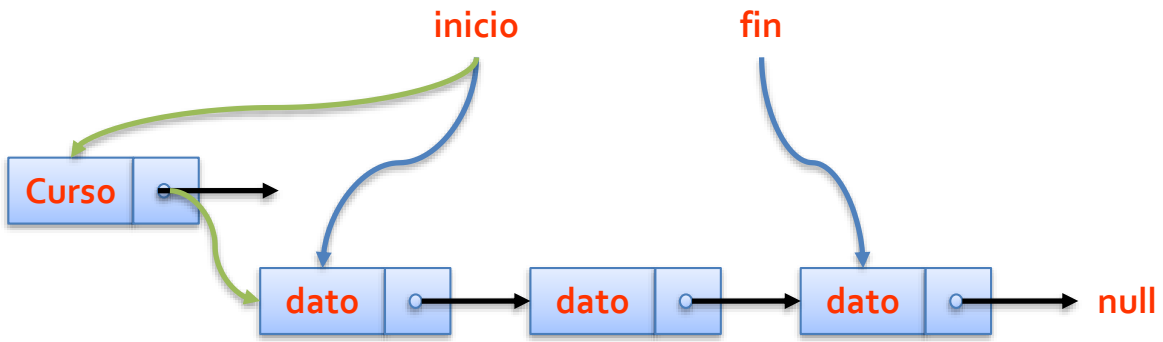

```java
public class ListaEnlazadaDR {
  private Nodo inicio;
  private Nodo fin;

  public ListaEnlazadaDR() {
    super();
    this.inicio = null;
    this.fin = null;
  }
}
```





Por comodidad se lo representa de esta manera

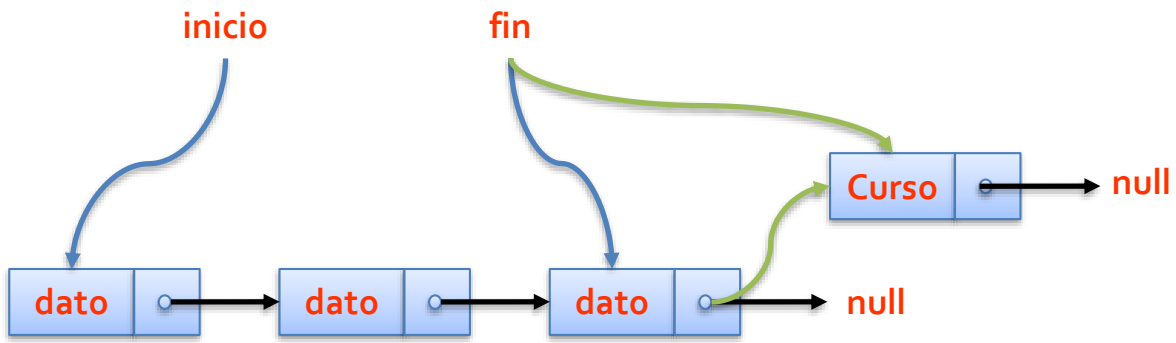
INSERTA UN NODO AL INICIO



return fin == null

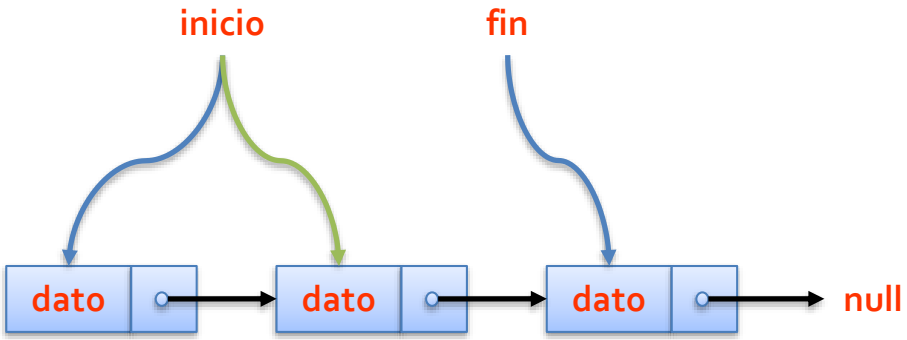
- Se parte de una lista no vacía
  - Se crea el Nodo new
    - `Nodo()`
  - Se inicializa el campo dato
    - `new Nodo(dato)`
  - El campo siguiente referencia al inicio de la lista
    - `new Nodo(dato, inicio)`
  - Se actualiza el inicio de la lista
    - `inicio = new Nodo(dato, inicio)`
- Si la lista está vacía
  - Se inserta el nodo pero también se hace que:
    - fin = inicio

INSERTA UN NODO AL FINAL



- Se parte de una lista no vacía
  - Se crea el Nodo new
    - `Nodo()`
  - Se inicializa el campo dato y que siguiente apunte a null
    - `new Nodo(dato)`
  - El sgt del Nodo final apunta al nuevo nodo
    - `Fin.setSgt(New Nodo(dato))`
  - Se actualiza el final de la lista
    - `fin = fin.getSgt()`
- Si la lista está vacía
  - Se inserta el nodo pero también se hace que:
    - Inicio = fin = new Nodo(dato)

ELIMINA EL NODO INICIAL

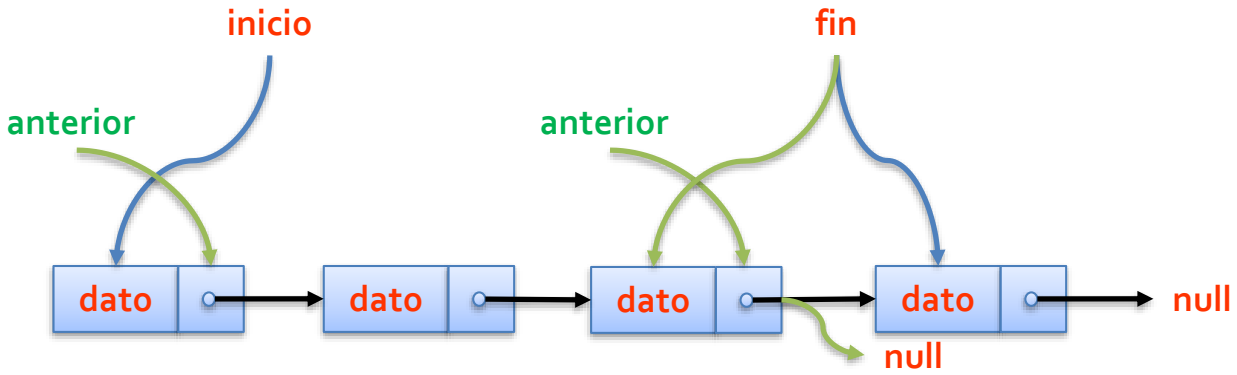



```
Curso curso = inicio.getDato()    //guarda el elemento a eliminar
```

```
inicio = fin = null     //si la lista tiene un solo elemento
```

- Si se tiene una lista no vacía
  - Se guarda una copia del el dato a eliminar.
  - Si la lista tiene más de dos elementos, inicio debe referenciar al siguiente elemento
    - `inicio = inicio.getSgt()`
  - Si la lista tiene un solo elemento, cuando inicio es igual a fin, entonces tanto inicio como fin referencian a null
    - `inicio = fin = null`
  - Retornar el elemento a eliminar
    - `return curso`
- Si la lista está vacía
    - `return null`

ELIMINA UN NODO FINAL




```
Curso curso = inicio.getDato()    //guarda el elemento a eliminar
```

```
inicio = fin = null     //si la lista tiene un solo elemento
```

- Si se tiene una lista no vacía
  - Se guarda una copia del el dato a eliminar.
  - Si la lista tiene un solo elemento, inicio y fin referencian a null
    - `inicio = fin = null`
  - Si la lista tiene más de dos elementos, se necesita un nodo que referencie al penúltimo elemento (anterior). Acciones:
  - Del tipo Nodo se define anterior e inicializa en inicio
    - `Nodo anterior = inicio`
  - Mientras anterior.sgt sea diferente de fin, anterior que referencie al Nodo siguiente
    - `anterior = anterior.getSgt()`
  - anterior.sgt referencia a null
  - fin referencia a anterior

LISTAS ENLAZADAS DOS REFERENCIAS

- Implemente el método estaVacia()
  - `private boolean estaVacia() {…}`
- Adicionalmente hay que implementar el método que permite mostrar la lista. Intente cambiar el método de mostrar la lista por el método toString()
  - `public String toString() {…}`
- Finalmente escriba su clase de prueba verificando el funcionamiento de todos los métodos implementados

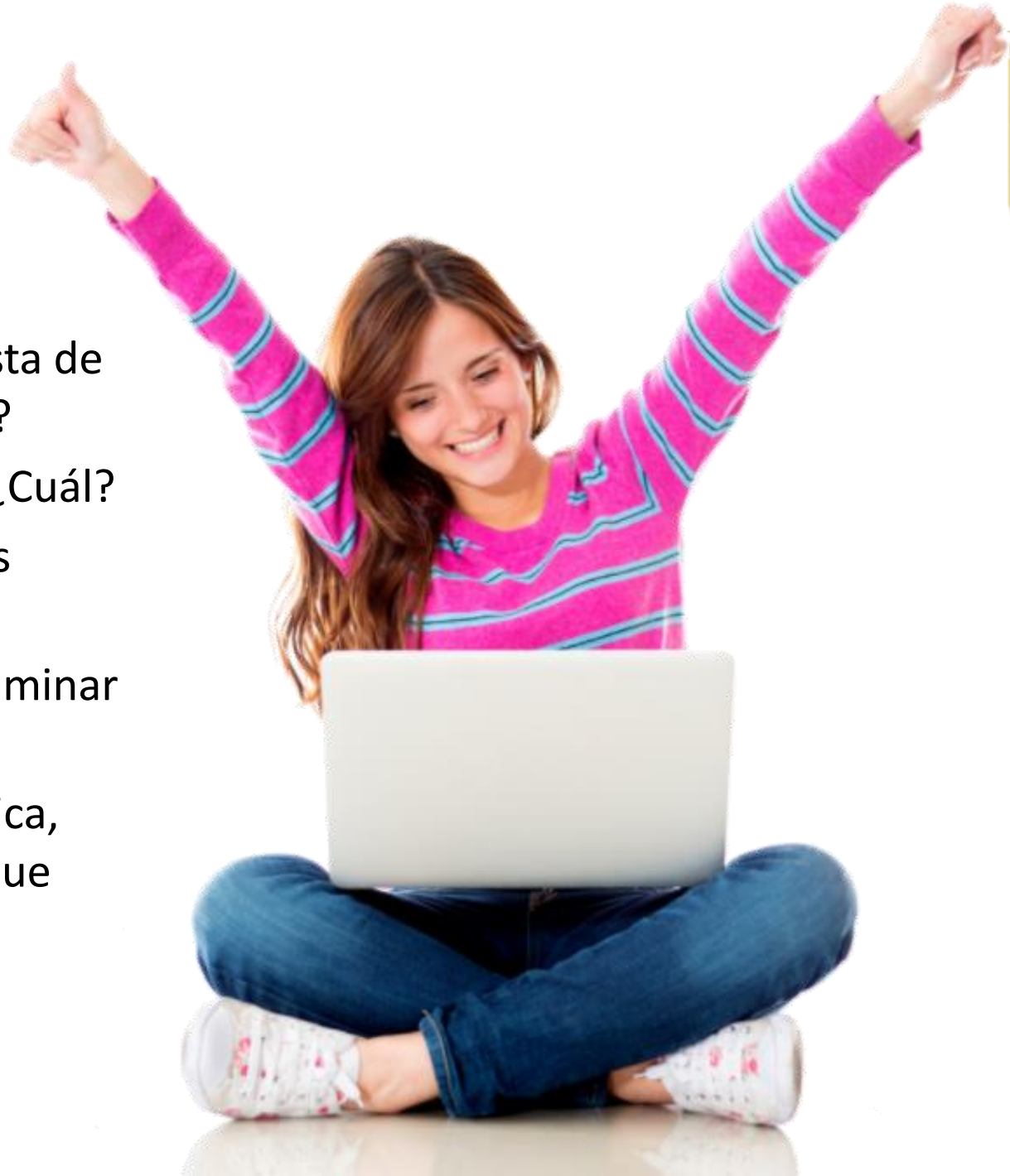
```
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    ListaEnlazadaDR listota = new ListaEnlazadaDR();
    System.out.println("Ingrese cantidad de elementos: ");
    int n = sc.nextInt();

    for (int i = 0; i < n; i++) {
        System.out.println("Dato "+(i+1)+": ");
        System.out.print("Ingrese nombre: ");
        String nombre = sc.next();
        System.out.print("Ingrese creditos: ");
        int creditos = sc.nextInt();

        Curso cursito = new Curso(nombre, creditos);

        listota.insertaFin(cursito);
    }
    System.out.println(listota);
    System.out.println("eliminando a ... " + listota.eliminaFin());
    System.out.println("eliminando a ... " + listota.eliminaFin());
    System.out.println(listota);
}
```

EVALUACIÓN DEL TEMA DESARROLLADO

Reflexionemos sobre lo aprendido!

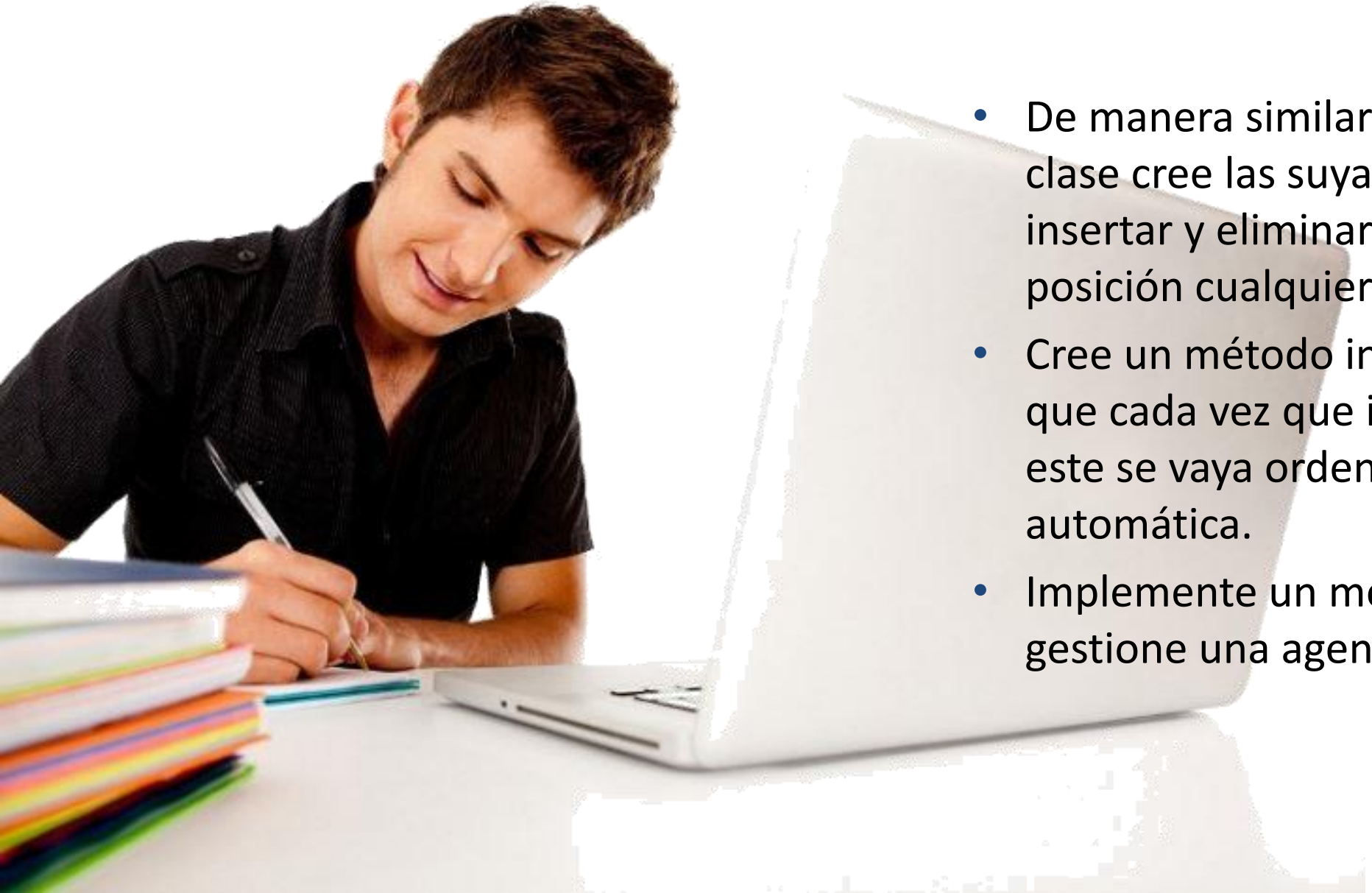
PREGUNTAS DE REPASO

- ¿Cómo solucionó el problema para insertar un elemento al final de la lista de tal modo que no recorra toda la lisa?
- ¿Observa todavía algún problema? ¿Cuál?
- ¿Qué ventajas tiene trabajar con dos referencias?
- Describa los pasos para insertar y eliminar un nodo
- Si quisiera crear una agenda telefónica, describa en un diagrama las clases que usaría



EJERCICIOS DE APLICACIÓN

EJERCICIOS DE APLICACIÓN



- De manera similar a las diapositivas de clase cree las suyas explicando como insertar y eliminar un Nodo en una posición cualquiera de la lista
- Cree un método insertaEnOrden(dato) que cada vez que inserte un elemento este se vaya ordenando de manera automática.
- Implemente un menú de opciones que gestione una agenda telefónica