Ejercicio U6_B2_E1:

```
class Persona{
    String nombre;
    int edad;

    Persona(String nombre,int edad){
        this.nombre=nombre;
        this.edad=edad;
    }
}

class Pair<T, V> {
    private T first;
    private V second;

    public T getFirst() {
        return first;
    }
    public V getSecond() {
        return second;
    }
    public void setFirst(T first) {
        this.first = first;
    }
    public void setSecond(V second) {
        this.second = second;
    }
}


public class App {

    public static void main(String[] args) {
        Pair<Integer,Float> parDeIntegerFloat=new Pair<>();
        parDeIntegerFloat.setFirst(4);
        parDeIntegerFloat.setSecond(9.8f);

        int[] a= {1,2,3};
        Pair<String, int[]> parDeStringArray=new Pair<>();
        parDeStringArray.setFirst("hola");
        parDeStringArray.setSecond(a);


        Persona p1= new Persona("Elías",5);
        Persona p2= new Persona("Román",4);
        Pair<Persona, Persona> parDePersona=new Pair<>();
        parDePersona.setFirst(p1);
        parDePersona.setSecond(p2);

    }
}
```

Ejercicio U6_B2_E2:

```
Pair(T t, V v){
    first=t;
    second=v;
  }
```

```
  T getFirst() {
    return first;
  }

  V getSecond() {
    return second;
  }

  void setFirst(T first) {
    this.first = first;
  }

  void setSecond(V second) {
    this.second = second;
  }
}

class Persona{
    String nombre;
    int edad;

    Persona(String nombre,int edad){
        this.nombre=nombre;
        this.edad=edad;
    }
}

class App {
    public static void main(String[] args) {
        Pair<Integer,Float> parDeIntegerFloat=new Pair<Integer,Float>(4,9.8f);


        int[] a= {1,2,3};
        Pair<String, int[]> parDeStringArray=new Pair<String,int[]>("hola",a);


        Persona p1= new Persona("Elías",5);
        Persona p2= new Persona("Román",4);
        Pair<Persona, Persona> parDePersona=new Pair<Persona,Persona>(p1,p2);

    }
}
```

Ejercicio U6_B2_E3:

```
 class Util {
    static <T> void imprimirArray (T[] t) {
        for(int i=0;i<t.length;i++)
            System.out.println(t[i].toString());
    }
}
class Persona{
    String nombre;
    int edad;

    Persona(String nombre,int edad){
        this.nombre=nombre;
        this.edad=edad;
    }
}
```

```
class App {

    public static void main(String[] args) {

        Persona p1= new Persona("Elías",5);
        Persona p2= new Persona("Román",4);
        Integer[] ArrayDeInteger={1,2,3};
        Double[] ArrayDeDouble={2.4,5.6,7.8};
        Persona[] ArrayDePersona={p1,p2};

        Util.<Integer>imprimirArray(ArrayDeInteger);
        Util.<Double>imprimirArray(ArrayDeDouble);
        Util.<Persona>imprimirArray(ArrayDePersona);

    }
}
```

Dos observaciones:

- La impresión del array de persona es un poco rara ya que el método toString() es heredado de la clase Object. Si queremos que imprima algo con sentido necesitaría ser redefinido en la clase persona. Por ejemplo,

```
public String toString(){
      return nombre+", "+edad;
}
```

- Cuando creamos el Array de Integer y Double estamos utilizando autoboxing ¿Recuerdas?

Ejercicio U6_B2_E4:

Javac sin -XLint

```
C:\Users\donlo\Documents\proyectos visual studio code\java\MiApp>cd src

C:\Users\donlo\Documents\proyectos visual studio code\java\MiApp\src>javac App.java
Note: App.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\donlo\Documents\proyectos visual studio code\java\MiApp\src>
```

Si compilo con -XLint (X mayúscula) veo una serie de advertencias que pueden generar problemas en tiempo de ejecución. Los warnings en este caso son debido a que utilizo tipos crudos en mi código.

```
C:\Users\donlo\Documents\proyectos visual studio code\java\MiApp\src>javac App.java -Xlint
App.java:17: warning: [rawtypes] found raw type: Nodo
  private Nodo sig;
          ^
  missing type arguments for generic class Nodo<T>
  where T is a type-variable:
    T extends Object declared in class Nodo
App.java:23: warning: [rawtypes] found raw type: Nodo
  public Nodo(T dato, Nodo sig) {
                      ^
  missing type arguments for generic class Nodo<T>
```

Corrijo evitando tipos crudos.

```
class Persona{
    String nombre;
    int edad;

    Persona(String nombre,int edad){
        this.nombre=nombre;
        this.edad=edad;
    }
    @Override
    public String toString(){
        return   nombre+", "+edad;
    }

}

class Nodo<T>{
  private Nodo<T> sig;
  private T dato;
  public Nodo(T dato) {
    this.dato = dato;
    this.sig=null;
  }
  public Nodo(T dato, Nodo<T> sig) {
    this.dato = dato;
    this.sig = sig;
  }
  public void setSiguiente(Nodo<T> sig) {
    this.sig = sig;
  }
  public Nodo<T> getSiguiente() {
    return sig;
  }
  public T getDato() {
    return dato;
  }
}
class MiListaEnlazada<T>{
    private Nodo<T> primero=null;
    public void insertar(T dato){
        if(primero==null){
            primero=new Nodo<>(dato);
        }else{
            Nodo<T> temp= new Nodo<>(dato,primero);
            primero=temp;
        }

    }
    public int tamano(){
        int i=0;
        Nodo<T> temp=primero;
        while(temp!=null){
            i++;
            temp=temp.getSiguiente();
        }
        return i;
    }
    public T obtener(int indice){
        Nodo<T> temp=primero;
        int i=0;
        while(i<indice){
            temp=temp.getSiguiente();
            i++;
        }
```

```
        return temp.getDato();
    }
}



public class App {

    public static void main(String[] args) {


        MiListaEnlazada<Persona> lp= new MiListaEnlazada<>();
        lp.insertar(new Persona("yo",23));
        lp.insertar(new Persona("tu",24));
        lp.insertar(new Persona("el",25));

        for(int i=0;i<lp.tamano();i++)
            System.out.print(lp.obtener(i)+"  ");

        MiListaEnlazada<Integer> li= new MiListaEnlazada<>();
        li.insertar(4);li.insertar(5);li.insertar(6);

       System.out.println();
        for(int i=0;i<li.tamano();i++)
            System.out.print(li.obtener(i)+"  ");
    }
}
```

Y ahora advierto que la compilación con Xlint es O.K.