Ejercicio U6_B2A_E1:

```
 interface Pila{
    //inserta un elemento en la cabeza de la pila
    void push(int dato);

    //saca un elemento de la cabeza de la pila.
    int pop();
    boolean esVacia() ;
}
class Nodo {
    private Nodo sig;
    private int dato;

    public Nodo(int dato) {
        this.dato = dato;
        this.sig = null;
    }

    public Nodo(int dato, Nodo sig) {
        this.dato = dato;
        this.sig = sig;
    }

    public void setSiguiente(Nodo sig) {
        this.sig = sig;
    }

    public Nodo getSiguiente() {
        return sig;
    }

    public int getDato() {
        return dato;
    }
}

class MiPila implements Pila{

    private Nodo cabeza = null;
    public void push(int dato) {
        if (cabeza == null) {
            cabeza = new Nodo(dato);
        } else {
            Nodo temp = new Nodo(dato, cabeza);
            cabeza = temp;
        }

    }

    //se puede hacer más simple el metodo push
    //esto funciona aunque la pila esté vacia
    //ya que new Nodo(dato,null) es equivalente a new Nodo(dato)
    public void push2(int dato){
        cabeza=new Nodo(dato,cabeza);
    }


    public int pop() {
        int dato = cabeza.getDato();
        cabeza = cabeza.getSiguiente();
        return dato;
    }

    public boolean esVacia() {
        return cabeza == null;
    }

}

class App {
    public static void main(String[] args) {
        Pila mipila = new MiPila();
        mipila.push(1);
        mipila.push(2);
        mipila.push(3);
        mipila.push(4);
        mipila.push(5);
        while (!mipila.esVacia()) {
            System.out.println(mipila.pop());
        }
    }
}
```

Ejercicio U6_B2A_E2:

```
interface Pila{
    //inserta un elemento en la cabeza de la pila
    void push(char dato);

    //saca un elemento de la cabeza de la pila.
    char pop();
   boolean esVacia() ;
}
class Nodo {
    private Nodo sig;
    private char dato;

    public Nodo(char dato, Nodo sig) {
        this.dato = dato;
        this.sig = sig;
    }

    public void setSiguiente(Nodo sig) {
        this.sig = sig;
    }

    public Nodo getSiguiente() {
        return sig;
    }

    public char getDato() {
        return dato;
    }
}

class MiPila implements Pila{

    private Nodo cabeza = null;
    public void push(char dato) {
        if (cabeza == null) {
            cabeza = new Nodo(dato,null);
        } else {
            Nodo temp = new Nodo(dato, cabeza);
            cabeza = temp;
        }

    }

    public char pop() {
        char dato = cabeza.getDato();
        cabeza = cabeza.getSiguiente();
        return dato;
    }

    public boolean esVacia() {
        return cabeza == null;
    }

}

class App {
   static boolean parentesisBalanceados(String expresion){
        MiPila p= new MiPila();
        boolean balanceados=true;
        for(char c: expresion.toCharArray()){
            if (c=='('){
                p.push(c);
            }else if (c==')'){
                if(p.esVacia()){
                    balanceados=false;
                    break;
                }else{
                    p.pop();
                }
            }
        }
        if(!p.esVacia()){
            balanceados=false;
        }
        return balanceados;
    }

    public static void main(String[] args) {
        String expresion="((2+3)/(3*(8-2))";
        System.out.println(parentesisBalanceados(expresion));
        expresion=")4(";
        System.out.println(parentesisBalanceados(expresion));
        expresion="(4)";
        System.out.println(parentesisBalanceados(expresion));
        expresion="(2+3)/(3*(8-2))";
        System.out.println(parentesisBalanceados(expresion));
```

```
    }
}
```

Ejercicio U6_B2A_E3: Cada uno tendrá su solución. Pero básicamente
PARA CADA CASO DE PRUEBA:

con %10 vamos extrayendo el último dígito y lo vamos metiendo en la pila.

una vez que tengo todos los dígitos en la pila la voy vaciando y al mismo tiempo generando el string de salida y calculando la suma total

Ejercicio U6_B2A_E4:

```
interface Cola {

    //inserta un elemento al final de la cola
    void encolar(int dato);

    //saca el primer elemento de la cola
    //el primer elemento es el más antiguo
    int desencolar();

    public boolean esVacia();
}

class Nodo {

    private Nodo sig;
    private int dato;



    public Nodo(int dato, Nodo sig) {
        this.dato = dato;
        this.sig = sig;
    }

    public void setSiguiente(Nodo sig) {
        this.sig = sig;
    }

    public Nodo getSiguiente() {
        return sig;
    }

    public int getDato() {
        return dato;
    }
}

class MiCola implements Cola {

    private Nodo primero = null;
    private Nodo ultimo = null;

    public boolean esVacia() {
        //vacia si primero==ultimo==null
        return ultimo == null;
    }

    @Override
    public void encolar(int dato) {
        if (esVacia()) {
            ultimo = new Nodo(dato,null);
            primero = ultimo;
        } else {
            Nodo temp = new Nodo(dato, ultimo);
            ultimo = temp;
        }

    }

    @Override
    public int desencolar() {
        //suponemos que no se invoca a desencolar con cola vacía
        int dato = primero.getDato();
        //recorrer la cola para hacer el segundo el primero

        if (primero == ultimo) {//o está vacía o tiene un elemento
            //si sólo hay un elemento al borrar la cola queda vacía
            ultimo = null;
```

```
            primero = null;
        } else {//al menos hay dos elementos
            Nodo temp = ultimo;
            while (temp.getSiguiente() != primero) {
                temp = temp.getSiguiente();
            }
            primero = temp;
            primero.setSiguiente(null);
        }

        return dato;
    }

}

class App {

    public static void main(String[] args) {
        MiCola mc = new MiCola();
        mc.encolar(1);
        mc.encolar(2);
        mc.encolar(3);
        while (!mc.esVacia()) {
            System.out.println(mc.desencolar());
        }
    }
}
```

Ejercicio U6_B2A_E5:

```
class Nodo {

    private Nodo sig;
    private int dato;

    public Nodo(int dato) {
        this.dato = dato;
        this.sig = null;
    }

    public Nodo(int dato, Nodo sig) {
        this.dato = dato;
        this.sig = sig;
    }

    public void setSiguiente(Nodo sig) {
        this.sig = sig;
    }

    public Nodo getSiguiente() {
        return sig;
    }

    public int getDato() {
        return dato;
    }
}

class MiListaEnlazada {

    private Nodo primero = null;

    public void insertar(int dato) {
        if (primero == null) {
            primero = new Nodo(dato);
        } else {
            Nodo temp = new Nodo(dato, primero);
            primero = temp;
        }

    }

    public void imprimirConFor() {
        System.out.println("imprimirConFor .....");
        for (Nodo temp = primero; temp != null; temp = temp.getSiguiente()) {
            System.out.print(temp.getDato() + " ");
        }
        System.out.println();
    }
    public void imprimirInvertidaConFor() {
```

```
        System.out.println("imprimirInvertidaConFor .....");
        //truco de crea lista temporal insertanto ahora al revés
        MiListaEnlazada listaTemp= new MiListaEnlazada();
        for (Nodo temp = primero; temp != null; temp = temp.getSiguiente()) {
            listaTemp.insertar(temp.getDato());
        }
        for (Nodo temp = listaTemp.primero; temp != null; temp = temp.getSiguiente()) {
            System.out.print(temp.getDato() + " ");
        }
        System.out.println();
    }

    public void imprimirConRec(){
        System.out.println("imprimirConRec.....");
        ayudanteImprimirConRec(primero);
        //imprimirConRec(primero);
        System.out.println();

    }

    private void ayudanteImprimirConRec(Nodo nodo){
        if(nodo!=null){
            System.out.print(nodo.getDato() + " ");
            ayudanteImprimirConRec(nodo.getSiguiente());
        }
    }
    /*
    * por  sobrecarga,  en lugar de ayudanteImprimirConRec
    *puedo usar también el nombre imprimirConRec
    *si es  private el programador de App "no lo ve"
    *y no lo confunde
    */
    private void imprimirConRec(Nodo nodo){
        if(nodo!=null){
            System.out.print(nodo.getDato() + " ");
            ayudanteImprimirConRec(nodo.getSiguiente());
        }
    }
    public void imprimirInvertidaConRec(){
        System.out.println("imprimirInvertidaConRec.....");
        ayudanteImprimirInvertidaConRec(primero);
        System.out.println();

    }
    private void ayudanteImprimirInvertidaConRec(Nodo nodo){
        if(nodo!=null){
            ayudanteImprimirInvertidaConRec(nodo.getSiguiente());
            System.out.print(nodo.getDato() + " ");
        }
    }
}

class App {

    public static void main(String[] args) {
        MiListaEnlazada miLista = new MiListaEnlazada();
        miLista.insertar(8);
        miLista.insertar(88);
        miLista.insertar(888);
        miLista.imprimirConFor();
        miLista.imprimirInvertidaConFor();
        miLista.imprimirConRec();
        miLista.imprimirInvertidaConRec();
    }
}
```