Actividades de aplicación. Realizar las siguientes 7.11, 7.12, 7.13, 7.14, 7.15, 7.16, 7.17, 7.18, y 7.19.

7.11. Escribe la clase `MarcaPagina`, que ayuda a llevar el control de la lectura de un libro. Deberá disponer de métodos para incrementar la página leída, para obtener información de la última página que se ha leído y para comenzar desde el principio una nueva lectura del mismo libro.

```java
package com.mycompany.aa7_11;

public class Aa7_11 {
    public static void main(String[] args) {
        Marcapagina marca;
        marca = new Marcapagina();
        marca.siguientePag();
        marca.siguientePag();
        marca.siguientePag();
        System.out.println("Última página: " + marca.ultimaPaginaLeida());
        marca.comenzar();
        System.out.println("Última página: " + marca.ultimaPaginaLeida());
    }
}
```

```java
package com.mycompany.aa7_11;

public class Marcapagina {
    private int pag = 0;

    public int ultimaPaginaLeida() {
        return pag;
    }

    public void setPag(int pag) {
        this.pag = pag;
    }

    void siguientePag(){
        pag++;
    }

    void comenzar() {
        pag = 0;
    }
}
```

```
Output - Run (Aa7_11)
Scanning for projects...

------------------------< com.mycompany:Aa7_11 >------------------------
Building Aa7_11 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ Aa7_11 ---
skip non existing resourceDirectory C:\Users\abrah\Documents\NetBeansProjects\Aa7_11\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ Aa7_11 ---
Changes detected - recompiling the module! :source
Compiling 2 source files with javac [debug target 21] to target\classes

--- exec:3.1.0:exec (default-cli) @ Aa7_11 ---
Última página: 3
Última página: 0
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.279 s
```

7.12. Implementa una clase que permita resolver ecuaciones de segundo grado. Los coeficientes pueden indicarse en el constructor y modificarse *a posteriori*. Es fundamental que la clase disponga de un método que devuelva las distintas soluciones y de un método que nos informe si el discriminante es positivo.

```java
package com.mycompany.aa7_12;

public class Ecuacion2Grado {
    private double a, b, c, discriminante;


    //Constructor
    public Ecuacion2Grado(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
        discriminante =  b * b - 4 * a * c;
    }

    //getter & setter
    public double getA() {
        return a;
    }

    public void setA(int a) {
        this.a = a;
        discriminante = b * b - 4 * a * c;
    }

    public double getB() {
        return b;
    }

    public void setB(int b) {
        this.b = b;
        discriminante = b * b - 4 * a * c;
    }

    public double getC() {
        return c;
    }

    public void setC(int c) {
        this.c = c;
        discriminante = b * b - 4 * a * c;
    }

    //Métodos
    String esPositivoDiscriminante() {
        return "El discriminante es: " + (discriminante >= 0 ? "positivo" : "negativo");
    }

    double[] solucion() { //x = [-b ± sqrt(b^2 - 4ac)] / (2a)
        double soluciones[] =  new double[2];
        if (discriminante >= 0) {
            soluciones[0] = (-b + Math.sqrt(a: discriminante)) / (2 * a);
            soluciones[1] = (-b - Math.sqrt(a: discriminante)) / (2 * a);
        }else {
             soluciones[0] = -b / (2 * a);
             soluciones[1] =  Math.sqrt(-discriminante) / (2 * a);
        }
        return soluciones;
    }


}
```

```
--- exec:3.1.0:exec (default-cli) @ Aa7_12 ---
El discriminante es: positivo
x1: 2.0
x2: 1.0
x1: -0.5
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

7.13. En el momento de decorar una casa, una habitación o cualquier objeto, se plantea el problema de elegir la paleta de colores que vamos a utilizar en nuestra decoración. Existe una solución, algo atrevida, que consiste en utilizar colores al azar.

Diseña la clase `Colores`, que alberga por defecto una serie de colores (mediante una cadena), aunque es posible añadir tantos como necesitemos. La clase tendrá un método que devuelve una tabla con los *n* colores que necesitemos elegidos al azar sin repeticiones.

```java
package com.mycompany.aa17_13;

public class Colores {
    private String[] colores = new String[0];

    //Métodos
    public void addColor(String color){
        String[] aux  = new String[colores.length + 1];
        System.arraycopy(src:colores, srcPos: 0, dest:aux, destPos:0, length: colores.length);
        aux[colores.length] = color;
        this.colores = aux;
        aux = null;
    }

    public String[] seleccionColores(int cantidad){
        String[] aux  = new String[cantidad];
        System.arraycopy(src:colores, srcPos: 0, dest:aux, destPos:0, length: cantidad);
        return aux;
    }
}
```

```
--- exec:3.1.0:exec (default-cli) @ Aa17_13 ---
[Marrón, Azul, Amarillo]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.426 s
```

7.14. Crea una clase que sea capaz de mostrar el importe de un cambio, por ejemplo, al realizar una compra, con el menor número de monedas y billetes posibles.

```java
package com.mycompany.aa7_14;

import java.math.BigDecimal;

public class Cambio {
    private double importe;
    private final double[] BILLETES = new double[]{5, 10, 20, 50, 100, 200, 500};
    private final double[] MONEDAS = new double[]{0.01, 0.02, 0.05, 0.1, 0.2,
        0.5, 1, 2};

    //Constructor
    public Cambio(double importe) {
        this.importe = importe;
    }

    /*ESTE MÉTODO PRODUCE UN PROBLEMA DE PRESICION AL USAR UN DOUBLE
        PARA RESOLVERLO HAY QUE USAR LA CLASE BigDecimal.*/
     public void mostrarCambio() {
        BigDecimal restante = BigDecimal.valueOf(val:importe);
        int i = 6;
        System.out.println("\nImporte a devolver: " + importe);
        System.out.println(x: "------Cambio a entregar------");
        while (restante.compareTo(val:BigDecimal.valueOf(val:5)) >= 0 && i >= 0) {
            BigDecimal[] result = restante.divideAndRemainder( divisor:BigDecimal.valueOf(BILLETES[i]));
            int cantidad = result[0].intValue();
            if (cantidad > 0) {
                System.out.println("Billetes de " + BILLETES[i] + ": " + cantidad);
                restante = result[1];
            }
            i--;
        }
        i = 7;
        while (restante.compareTo(val:BigDecimal.ZERO) > 0 && i >= 0) {
            BigDecimal[] result = restante.divideAndRemainder( divisor:BigDecimal.valueOf(MONEDAS[i]));
            int cantidad = result[0].intValue();
            if (cantidad > 0) {
                System.out.println("Monedas de " + MONEDAS[i] + ": " + cantidad);
                restante = result[1];
            }
            i--;
        }
    }
```

```
Importe a devolver: 12.3
------Cambio a entregar------
Billetes de 10.0: 1
Monedas de 2.0: 1
Monedas de 0.2: 1
Monedas de 0.1: 1

Importe a devolver: 1234.56
------Cambio a entregar------
Billetes de 500.0: 2
Billetes de 200.0: 1
Billetes de 20.0: 1
Billetes de 10.0: 1
Monedas de 2.0: 2
Monedas de 0.5: 1
Monedas de 0.05: 1
Monedas de 0.01: 1
------------------------------------------
BUILD SUCCESS
------------------------------------------
```

7.15. Diseña la clase `Calendario` que representa una fecha concreta (año, mes y día). La clase debe disponer de los métodos:

- `Calendario(int año, int mes, int dia)`: que crea un objeto con los datos pasados como parámetros, siempre y cuando, la fecha que representen sea correcta.
- `void incrementarDia()`: que incrementa en un día la fecha del calendario.
- `void incrementarMes()`: que incrementa en un mes la fecha del calendario.
- `void incrementarAño(int cantidad)`: que incrementa la fecha del calendario en el número de años especificados. Ten en cuenta que el año 0 no existió.
- `void mostrar()`: muestra la fecha por consola.
- `boolean iguales(Calendario otraFecha)`: que determina si la fecha invocante y la que se pasa como parámetro son iguales o distintas.

Por simplicidad, solo tendremos en consideración que existen meses con distinto número de días, pero no tendremos en cuenta los años bisiestos.

```java
package com.mycompany.aa7_15;

public class Calendario {
    private int dia, mes, año;

    public Calendario(int dia, int mes, int año) {
//Entiendo que año<0 = A.C.----También obvio el nº de dias de cada mes concreto
        if (dia > 0 && dia < 32 && mes > 0 && mes < 13 && año != 0) {
            this.dia = dia;
            this.mes = mes;
            this.año = año;
        }//en caso contrario lanzaría una excepción pero no lo hemos visto aún
    }

    void mostrar() {
        System.out.println("Fecha: " + dia  + '/' + mes + '/' + año);
    }

    void incrementarDia() {
        if(dia == 31){
            incrementaMes();
            dia = 1;
        }else{
            dia ++;
        }
    }

    void incrementaMes() {
        if (mes == 12) {
            incrementaAño();
            mes = 1;
        } else {
            mes++;
        }
    }

    void incrementaAño() {
        año ++;
    }

    String iguales(Calendario otro){
        return (otro.año == año) && (otro.mes == mes) && (otro.dia == dia) ? "SI" : "NO";
    }
}
```

```
--- exec:3.1.0:exec (default-cli) @ Aa7_15 ---
Fecha: 31/12/2021
Fecha: 1/1/2022
Fecha: 1/2/2022
Iguales: SI
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

7.16. Escribe la clase `Punto` que representa un punto en el plano (con un componente x y un componente y), con los métodos:

- `Punto(double x, double y)`: construye un objeto con los datos pasados como parámetros.
- `void desplazaX(double dx)`: incrementa el componente x en la cantidad `dx`.
- `void desplazaY(double dy)`: incrementa el componente y en la cantidad `dy`.
- `void desplaza(double dx, double dy)`: desplaza ambos componentes según las cantidades `dx` (en el eje x) y `dy` (en el componente y).
- `double distanciaEuclidea(Punto otro)`: calcula y devuelve la distancia euclídea entre el punto invocante y el punto `otro`.
- `void muestra()`: muestra por consola la información relativa al punto.

```java
package com.mycompany.aa7_16;

public class Aa7_16 {
    public static void main(String[] args) {
        Punto p1 = new Punto(x: 1, y: 4);
        p1.desplazaX(dX: 1);
        p1.desplazaY(dY: -2);
        p1.muestra();
        Punto p2 = new Punto(x: 3, y: 3);

        System.out.println("Distancia: " + p1.distanciaEuclidea(p2));
    }

}
```

```java
package com.mycompany.aa7_16;

public class Punto {
    private double x, y;

    //constructor
    public Punto(double x, double y) {
        this.x = x;
        this.y = y;
    }

    //Métodos
    public void desplazaX(double dX){
        x += dX;
    }

    public void desplazaY(double dY){
        y += dY;
    }

    public void desplaza(double dX, double dY){
        x += dX;
        y += dY;
    }

    public double distanciaEuclidea(Punto p2){
        return Math.sqrt( Math.pow(p2.getX() - x, b: 2)
                + Math.pow(p2.getY() - y, b: 2));
    }

    public void muestra(){
        System.out.println("El punto se encuentra en las coordenadas X:" + x +
                " Y:" + y);
    }

    //getter
    public double getX() {return x;}
    public double getY() {return y;}
}
```

```
--- exec:3.1.0:exec (default-cli) @ Aa7_16 ---
El punto se encuentra en las coordenadas X:2.0 Y:2.0
Distancia: 1.4142135623730951
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.602 s
```

7.17. El cifrado César es una forma sencilla de modificar un texto para que no sea entendible a quienes no conocen el código. Este cifrado consiste en modificar cada letra de un texto por otra que se encuentra en el alfabeto *n* posiciones detrás.

Por ejemplo, para un valor de *n* igual a 3, la letra *a* se codifica con la *d*, y la letra *q* se codifica con la *x*. En el caso de que una letra exceda a la *z*, seguiremos de forma circular utilizando la *a*. Solo se cifrarán las letras, mayúsculas o minúsculas.

Realiza una clase que, mediante un método estático, devuelva cifrado el texto que se le pasa con un paso de *n* letras.

```java
package com.mycompany.aa7_17;

public class Aa7_17 {
    public static void main(String[] args) {
        System.out.println(x: Cifrado.cesar(input: "abcdef ... vwxyz", paso: 3));
        System.out.println(x: Cifrado.cesar(input: "Hola mundo", paso: 5));
    }
}
```

```java
package com.mycompany.aa7_17;

public class Cifrado {

    public static String cesar(String input, int paso){
        char[] array = input.toCharArray();
        int i= 0;
        for(char letra: array){
            array[i] = Character.isLetter(letra + paso) ?  (char)(letra + paso)
                    : asignador(letra, paso);
            i++;
        }

        return new String(value: array);
    }

    private static char asignador(char letra, int paso){
        if(!Character.isLetter(ch: letra)){
            return letra;
        }else if ( Character.isLowerCase(ch: letra)){
            while( letra <= 'z' && paso > 0){
                letra++;
                paso--;
            }
            if( paso > 0){
                return (char)('a' + paso);
            }
            return 'a';
        }else{
             while( letra <= 'Z' && paso > 0){
                letra++;
                paso--;
            }
            if( paso > 0){
                return (char)('A' + paso);
            }
            return 'A';
        }
    }
}
```

```
--- exec:3.1.0:exec (default-cli) @ Aa7_17 ---
defghi ... yzabc
Mtqf rzsit
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

7.18. Una cola es otra estructura dinámica como la pila, donde los elementos, en vez de apilarse y desapilarse, se encolan y desencolan. La diferencia con las pilas es que se desencola el primer elemento encolado, ya que así es como funcionan las colas del autobús o del cine. El primero que llega es el primero que sale de la cola (vamos a suponer que nadie se cuela). Por tanto, los elementos se encolan y desencolan en extremos opuestos de la estructura, llamados *primero* (el que está primero y será el próximo en abandonar la cola) y *último* (el que llegó último). Implementa la clase `Cola` donde los elementos `Integer` encolados se guardan en una tabla.

```java
package com.mycompany.aa7_18;

public class Aa7_18 {
    public static void main(String[] args) {
        Cola c = new Cola();
        for (int i = 1; i <= 10; i++) {
            c.encola(num:i);
        }

        System.out.println("Primero: " + c.primero());
        System.out.println("Vacía: " + c.vacia());
        while(!c.vacia()) {
            System.out.println(x: c.desencola());
        }
    }
}
```

```java
package com.mycompany.aa7_18;

import java.util.Arrays;

public class Cola {
    int[] cola = new int[0];

    public void encola(int num){
        cola = Arrays.copyOf(original: cola, cola.length + 1);
        cola[cola.length - 1] = num;
    }
    //Entiendo mirando el main que devuelve el array en String
    public String desencola(){
        cola = Arrays.copyOfRange(original: cola, from: 1, to: cola.length);
        return Arrays.toString(a: cola);
    }

    public int primero(){
        return cola[0];
    }
    //Entiendo (mirando el main) que este método indica si está vacía o no
    public boolean vacia(){
        return cola.length == 0;
    }

}
```

```
Output - Run (Aa7_18)
--- exec:3.1.0:exec (default-cli) @ Aa7_18 ---
Primero: 1
Vacía: false
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[3, 4, 5, 6, 7, 8, 9, 10]
[4, 5, 6, 7, 8, 9, 10]
[5, 6, 7, 8, 9, 10]
[6, 7, 8, 9, 10]
[7, 8, 9, 10]
[8, 9, 10]
[9, 10]
[10]
[]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

7.19. Implementa la clase `Pila` para números `Integer`, usando directamente una tabla para guardar los elementos apilados.

```java
package com.mycompany.aa7_19;

public class Aa7_19 {
    public static void main(String[] args) {
        Pila p = new Pila();
        for (int i = 1; i <= 10; i++) {
            p.apila(num:i);
        }

        System.out.println("Cima: " + p.cima());
        while(!p.vacia()) {
            System.out.println(x: p.desapila());
        }
    }
}
```

```java
public class Pila {
    int[] pila = new int[0];

    public void apila(int num){
        int[] aux = new int[pila.length + 1];
        System.arraycopy(src:pila, srcPos: 0, dest:aux, destPos:1, length: pila.length);
        aux[0] = num;
        pila = aux;
    }
    //Entiendo mirando el main que devuelve el array en String
    public String desapila(){
        if (pila.length > 0) {
            pila = Arrays.copyOf(original:pila, pila.length - 1);
        }
        return Arrays.toString(a: pila);
    }

    public int cima(){
        return pila[pila.length - 1];
    }
    //Entiendo (mirando el main) que este método indica si está vacía o no
    public boolean vacia(){
        return pila.length == 0;
    }
}
```

Output - Run (Aa7_19)

```
Compiling 2 source files with javac [debug target 21] to target\classes

--- exec:3.1.0:exec (default-cli) @ Aa7_19 ---
Cima: 1
[10, 9, 8, 7, 6, 5, 4, 3, 2]
[10, 9, 8, 7, 6, 5, 4, 3]
[10, 9, 8, 7, 6, 5, 4]
[10, 9, 8, 7, 6, 5]
[10, 9, 8, 7, 6]
[10, 9, 8, 7]
[10, 9, 8]
[10, 9]
[10]
[]
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

Actividades de comprobación. Realizarlas todas. Copiar todas las preguntas y sus respuestas correctas.

7.1. Dos clases se consideran vecinas siempre y cuando:

a) Sean visibles.
b) Ambas dispongan del mismo número de constructores.
c) Pertenezcan al mismo paquete.
d) Todo lo anterior ha de cumplirse para que dos clases sean vecinas.

C

7.2. Un miembro cuyo modificador de acceso es `private` será visible desde:

a) Todas las clases vecinas.
b) Todas las clases externas.
c) Es indistinto el paquete, pero será visible siempre que se importe la clase que lo contiene.
d) Ninguna de las respuestas anteriores.

D

7.3. Si desde un constructor queremos invocar a otro constructor de la misma clase, tendremos que usar:

a) `set()`.
b) `get()`.
c) `this()`.
d) `this`.

C

7.4. **Si por error dejamos un objeto sin ninguna referencia, siempre podremos volver a referenciarlo mediante:**

a) La referencia `this`.
b) La referencia `null`.
c) Utilizando `new`.
d) Es imposible.

D

7.5. **¿Qué hace el operador `new`?**

a) Construye un objeto, invoca al constructor y devuelve su referencia.
b) Construye un objeto, comprueba que su clase esté importada y devuelve su referencia.
c) Busca en la memoria un objeto del mismo tipo, invoca al constructor y devuelve su referencia.
d) Busca en memoria un objeto del mismo tipo y devuelve su referencia.

A

7.6. **Cuando hablamos de miembros de una clase, nos estamos refiriendo a:**

a) Todos los atributos.
b) Todos los métodos.
c) Todos los atributos y métodos, indistintamente de los modificadores de acceso utilizados.
d) Todos los atributos y métodos que son visibles por sus clases vecinas.

C

7.7. **En la definición de una clase, los únicos modificadores de acceso que se pueden utilizar son:**

a) `public`.
b) `public` y el modificador de acceso por defecto.
c) `public`, el modificador de acceso por defecto y `private`.
d) El modificador `class`.

B

7.8. ¿Qué diferencia un atributo estático definido en una clase de otro que no lo es?

a) El atributo estático es visible por todas las clases vecinas, mientras que el no estático solo será visible para las clases que usen importación.

b) Solo existe una copia del atributo estático en la clase, mientras que el atributo no estático tendrá una copia en cada uno de los objetos.

c) Existe una copia del atributo estático en todos y cada uno de los objetos, mientras que del atributo no estático solo existe una copia en la clase.

d) Ambos disponen de copias en cada objeto, pero el atributo no estático es accesible mediante la clase y el no estático es accesible mediante los objetos.

B

7.9. ¿Qué efecto tiene las siguientes líneas de código?

```
Cliente c;
c.nombre = "Pepita";
```

a) Inicializa el atributo nombre de `Cliente` con el valor «Pepita».

b) Invoca al constructor y posteriormente asigna el valor «Pepita» al atributo nombre, siempre y cuando este sea público.

c) Si el atributo `nombre` es público, se le asigna un valor, pero si el atributo es privado, producirá un error.

d) Siempre produce un error.

D

7.10. La ocultación de atributos puede definirse como:

a) El proceso en el que un atributo pasa de ser público a privado.

b) El proceso en el que se define una variable local (en un método) con el mismo identificador que un atributo.

c) El proceso en el que un atributo estático deja de serlo.

d) Todas las respuestas anteriores son correctas.

B