Tarea online

Título de la tarea: Manejos de estructuras de datos internas.

Unidad: 07

Ciclo formativo y módulo: DAM/DAW, Programación.

Curso académico: 2025/2026

¿Qué contenidos o resultados de aprendizaje trabajaremos?

En esta tarea trabajaremos con el siguiente resultado de aprendizaje:

- RA6.- *Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos*.

En especial, en esta unidad se trabaja con tipos avanzados de datos: **listas**, **mapas** (o estructuras asociativas) y **conjuntos**, con distintos tipos de implementaciones internas (árboles, tablas hash, etc.), estructuras de datos extensamente usadas en programación.

1.- Descripción de la tarea

Caso práctico

Ministerio de Educación. Uso educativo-nc. Elaboración propia.

En **BK Programación** les ha surgido un proyecto muy interesante, pero es una aplicación en la que se usan masivamente diferentes tipos de colecciones, y esto a Juan le ha puesto un poco nervioso, dado que no recuerda muy bien cómo se trabajan con conjuntos, listas y mapas.

Juan decide comentárselo a **María**:

—**María**, ¿tú recuerdas para qué se utilizaba un mapa? Estoy un poco perdido.

—Sí claro, la idea es poder encontrar un valor entre un conjunto de valores partiendo de una clave. Las colecciones de Java son extremadamente potentes y útiles. Si no las usas estás perdido. —comenta **María** sonriendo.

—¿Cómo? Pues entonces sí que estoy perdido la verdad. —comenta **Juan**.

—No te preocupes, seguro que refrescando la memoria un poco podrás usarlas sin problemas.

—Ya veremos, intentaré practicar un poco en casa antes de empezar con el proyecto. —comenta **Juan** un poco resignado.

¿Qué te pedimos que hagas?

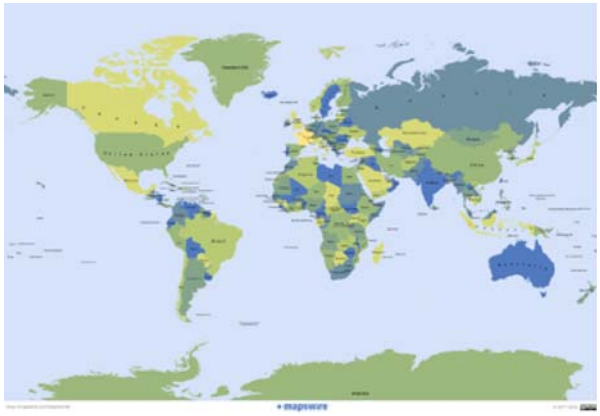
Esta tarea está compuesta por varios ejercicios en los que se trabaja con algunas de las estructuras de datos que has estudiado en la unidad. Recuerda que en cada ejercicio debes utilizar la estructura indicada en el mismo o bien la que consideres más apropiada si no se indica de manera explícita.

Encontrarás un proyecto base de NetBeans sobre el que trabajar en la sección 2.- Información de interés. Ese proyecto contiene herramientas específicas para la realización de esta tarea con algunas clases que tendrás que usar en algunos de los ejercicios (como por ejemplo la clase `Utilidades`).

Ten en cuenta que **en ningún ejercicio debes hacer uso de arrays para resolver los problemas**. No tendrás que rellenar ni generar arrays como parte de tu solución. La idea es que en esta tarea trabajes con estructuras de datos avanzadas de tipo dinámico (estilo `Set`, `List` o `Map`) y no estáticas (como serían los arrays, que ya empleaste en tareas anteriores).

1.1.- Ejercicio 1: creando conjuntos de países

Vamos a trabajar con conjuntos de `String` que contengan los países de la Unión Europea y de las mayores economías mundiales. Para generar estos conjuntos disponemos de una clase `"Utilidades"` donde se encuentran implementados los



mapswire. *Mapa* (Dominio público)

métodos `Utilidades.getARRAY_PAISES_UE()` y `Utilidades.getARRAY_PAISES_MUNDO()`, que nos devolverán un array de `String` con los países de la Unión Europea y de las mayores economías mundiales respectivamente.

El formato que sigue cada uno de los objetos de tipo `String` contenidos en los arrays es:

- **"1.70 - España (EUR)"**, donde:
  - **"1.70"**: se corresponde con el PIB del país expresado en billones de euros. Esta cantidad siempre tendrá el mismo formato (un número entero y dos números decimales);
  - **' - '**: se corresponde con la cadena separadora entre el PIB y el nombre del país;
  - **"España"**: se corresponde con el nombre del país;
  - **" ":** se corresponde con el carácter espacio y separa el nombre del país de la abreviatura del continente;
  - **"(EUR)"**: se corresponde con el continente en formato abreviado. Siempre tendrá el formato de 3 letras encerradas entre paréntesis.

Escribe un programa en Java que lleve a cabo las siguientes acciones:

1. Genere **dos conjuntos** de `String` (estructuras de tipo `Set`) con los países de la Unión Europea y con los de las mayores economías mundiales. Considera tipos de estructuras en las que cada país puede aparecer una única vez en cada conjunto.
2. Rellene un tercer conjunto con la **unión de los dos conjuntos** anteriores.
3. Rellene un cuarto conjunto con la **intersección de los dos conjuntos** anteriores.
4. Genere un quinto conjunto con la **diferencia del segundo conjunto menos el primero**, es decir, quitar al segundo conjunto, los elementos (países) que haya en el primero.
5. **Muestre por pantalla el contenido** de cada uno de esos conjuntos.

> Recuerda que si no se especifica de manera explícita el tipo de implementación para la estructura de datos (en este caso para un conjunto o `Set`), puedes elegir la que consideres más oportuna de las que haya disponibles (`TreeSet`, `HashSet`, etc.).

Revisa los **métodos disponibles en la interfaz** que implementa la estructura utilizada para optimizar el código y evitar realizar operaciones innecesarias.

Ejemplos de ejecución

Aquí tienes una muestra de cómo debería quedar el resultado final de la ejecución del programa:

Un ejemplo de ejecución podría ser el siguiente:

```
CONJUNTOS DE PAISES DE LA UE Y PAISES CON LAS MAYORES ECONOMÍAS MUNDIALES
--------------------------------------------------------------------------
Conjunto C1 (países UE):
1. 0.09 - Luxemburgo (EUR)
2. 3.20 - Francia (EUR)
3. 0.25 - Grecia (EUR)
4. 1.70 - España (EUR)
5. 0.38 - Rumanía (EUR)
6. 0.23 - Hungría (EUR)
7. 0.05 - Letonia (EUR)
8. 0.14 - Eslovaquia (EUR)
9. 0.30 - Portugal (EUR)
10. 0.05 - Estonia (EUR)
11. 0.56 - Irlanda (EUR)
12. 0.54 - Austria (EUR)
13. 0.09 - Lituania (EUR)
14. 0.61 - Suecia (EUR)
15. 0.86 - Polonia (EUR)
16. 0.11 - Bulgaria (EUR)
17. 0.31 - Finlandia (EUR)
18. 0.34 - República Checa (EUR)
19. 0.04 - Chipre (EUR)
20. 0.41 - Dinamarca (EUR)
21. 0.03 - Malta (EUR)
22. 0.09 - Croacia (EUR)
23. 0.66 - Bélgica (EUR)
24. 0.07 - Eslovenia (EUR)
25. 2.40 - Italia (EUR)
26. 1.20 - Países Bajos (EUR)
27. 4.70 - Alemania (EUR)

Conjunto C2 (países mayores economías):
1. 1.83 - Australia (OCE)
2. 8.50 - China (ASI)
3. 4.28 - Japón (ASI)
4. 3.20 - Francia (EUR)
5. 2.54 - Rusia (EUR)
6. 2.28 - Canadá (AME)
```

7. 2.26 - Brasil (AME)
8. 1.27 - Arabia Saudita (ASI)
9. 1.57 - Turquía (EUR)
10. 1.70 - España (EUR)
11. 1.44 - Indonesia (ASI)
12. 1.86 - Corea del Sur (ASI)
13. 1.04 - Polonia (EUR)
14. 9.90 - Estados Unidos (AME)
15. 3.96 - Reino Unido (EUR)
16. 1.86 - México (AME)
17. 2.40 - Italia (EUR)
18. 1.20 - Países Bajos (EUR)
19. 4.70 - Alemania (EUR)
20. 4.13 - India (ASI)

Unión C1 y C2:
1. 0.09 - Luxemburgo (EUR)
2. 1.83 - Australia (OCE)
3. 8.50 - China (ASI)
4. 3.20 - Francia (EUR)
5. 0.25 - Grecia (EUR)
6. 2.54 - Rusia (EUR)
7. 1.27 - Arabia Saudita (ASI)
8. 1.57 - Turquía (EUR)
9. 1.70 - España (EUR)
10. 0.38 - Rumanía (EUR)
11. 1.44 - Indonesia (ASI)
12. 0.23 - Hungría (EUR)
13. 0.05 - Letonia (EUR)
14. 0.14 - Eslovaquia (EUR)
15. 0.30 - Portugal (EUR)
16. 0.05 - Estonia (EUR)
17. 9.90 - Estados Unidos (AME)
18. 0.56 - Irlanda (EUR)
19. 3.96 - Reino Unido (EUR)
20. 4.13 - India (ASI)
21. 0.54 - Austria (EUR)
22. 0.09 - Lituania (EUR)
23. 0.61 - Suecia (EUR)
24. 4.28 - Japón (ASI)
25. 0.86 - Polonia (EUR)
26. 0.11 - Bulgaria (EUR)
27. 2.28 - Canadá (AME)
28. 2.26 - Brasil (AME)
29. 0.31 - Finlandia (EUR)
30. 0.34 - República Checa (EUR)
31. 0.04 - Chipre (EUR)
32. 0.41 - Dinamarca (EUR)
33. 0.03 - Malta (EUR)
34. 1.86 - Corea del Sur (ASI)
35. 1.04 - Polonia (EUR)
36. 0.09 - Croacia (EUR)
37. 0.66 - Bélgica (EUR)
38. 0.07 - Eslovenia (EUR)
39. 1.86 - México (AME)
40. 2.40 - Italia (EUR)
41. 1.20 - Países Bajos (EUR)
42. 4.70 - Alemania (EUR)

```
Interseccion C1 y C2:
1. 3.20 - Francia (EUR)
2. 1.70 - España (EUR)
3. 2.40 - Italia (EUR)
4. 1.20 - Países Bajos (EUR)
5. 4.70 - Alemania (EUR)

Diferencia C1 y C2:
1. 1.83 - Australia (OCE)
2. 8.50 - China (ASI)
3. 4.28 - Japón (ASI)
4. 2.54 - Rusia (EUR)
5. 2.28 - Canadá (AME)
6. 2.26 - Brasil (AME)
7. 1.27 - Arabia Saudita (ASI)
8. 1.57 - Turquía (EUR)
9. 1.44 - Indonesia (ASI)
10. 1.86 - Corea del Sur (ASI)
11. 1.04 - Polonia (EUR)
12. 9.90 - Estados Unidos (AME)
13. 3.96 - Reino Unido (EUR)
14. 1.86 - México (AME)
15. 4.13 - India (ASI)
```

Como podemos comprobar:

- En el conjunto de unión obtenemos todos los países.
- En el conjunto de intersección obtenemos los países comunes en ambos conjuntos.
- En el conjunto de diferencia obtenemos los módulos del segundo conjunto que no están en el primero.

Recomendación

Recuerda que a la hora de mostrar el contenido de las estructuras de datos avanzadas con su formato correspondiente tienes disponible el método **`mostrarPaises()`** de la clase **`Utilidades`**.

1.2.- Ejercicio 2: trabajando con listas de países

Implementa un programa en Java donde:

Ivan Samkov (CC0)

1. Se **rellenen dos estructuras de tipo lista** (`List`) con los **países** de la **Unión Europea y de las mayores economías mundiales** (objetos de la clase `String`). Para generar estas listas disponemos de las herramientas `Utilidades.getARRAY_PAISES_UE()` y `Utilidades.getARRAY_PAISES_MUNDO()`, que nos devolverán un array de `String` con los países correspondientes (siguiendo estos objetos `String` el formato indicado en el ejercicio 1). Una vez tengamos ambas listas rellenas, comenzaremos a trabajar con ellas.
2. Mediante la **lista** que contenga los **países de la Unión Europea** se generará otra lista en base a los países que que superen el billón de euros en su PIB. Se inspeccionará la lista que contenga los países de la Unión Europea y si el PIB del país supera el billón de euros:
    1. se almacenará ese país en la **lista de países más ricos de la UE**;
    2. se almacenará también ese país en un **tipo de colección apropiado** para conseguir que los países aparezcan una única vez;
    3. se "marcará" como país "rico" añadiendo dos '$' al comienzo y final del nombre del país en la **lista de países de la Unión Europea** que estamos inspeccionando.
3. Se ordenará la **lista** que contiene los **países con las mayores economías mundiales** conforme a los siguientes criterios:
    1. por **nombre** del país (alfabético ascendente);
    2. por **PIB** del país (numérico ascendente).
4. Para ello habrá que escribir el código de **dos clases** que **implementen** la **interfaz** `Comparator` que nos permitan ordenar esa lista en función de dos criterios diferentes:
    1. clase `ComparadorPaisesPorNombre;`
    2. clase `ComparadorPaisesPorPIB.`

    Puedes escribir las dos clases dentro del propio archivo del ejercicio, fuera de la clase principal (como clases no públicas), o bien implementarlas de manera independiente cada una en un archivo.
5. **Muestra** por pantalla:
    1. el **contenido inicial** de la **lista de países de la Unión Europea**;

2. el **contenido inicial** de la **lista de países con las mayores economías mundiales**;
3. la **lista de países con las mayores economías mundiales ordenada** utilizando el **comparador** por **nombre** de país;
4. la **lista de países con las mayores economías mundiales ordenada** utilizando el **comparador** por **PIB** del país;
5. el **contenido final** de la **lista de países de la Unión Europea** (con algunas posiciones "marcadas");
6. la **lista de países de la Unión Europea con el PIB superior al billón de euros**;
7. el **conjunto de países de la Unión Europea con el PIB superior al billón de euros**.

NOTA: Recuerda que para recorrer colecciones debes hacer uso de **iteradores**, siempre que sea posible.

Revisa los **métodos disponibles en la interfaz** que implementa la estructura utilizada para optimizar el código y evitar realizar operaciones innecesarias.

Ejemplos de ejecución

Aquí tienes una muestra de cómo debería quedar el resultado final de la ejecución del programa:

Aquí tienes un ejemplo de ejecución con las diferentes listas que se solicitan en el ejercicio, así como del conjunto.

```
LISTAS DE PAISES
----------------

Contenido inicial de la lista países de la UE:
1. 4.70 - Alemania (EUR)
2. 3.20 - Francia (EUR)
3. 2.40 - Italia (EUR)
4. 1.70 - España (EUR)
5. 1.20 - Países Bajos (EUR)
6. 0.86 - Polonia (EUR)
7. 0.66 - Bélgica (EUR)
8. 0.61 - Suecia (EUR)
9. 0.56 - Irlanda (EUR)
10. 0.54 - Austria (EUR)
11. 0.41 - Dinamarca (EUR)
12. 0.38 - Rumanía (EUR)
13. 0.34 - República Checa (EUR)
14. 0.31 - Finlandia (EUR)
15. 0.30 - Portugal (EUR)
16. 0.25 - Grecia (EUR)
17. 0.23 - Hungría (EUR)
```

18. 0.14 - Eslovaquia (EUR)
19. 0.11 - Bulgaria (EUR)
20. 0.09 - Luxemburgo (EUR)
21. 0.05 - Estonia (EUR)
22. 0.05 - Letonia (EUR)
23. 0.04 - Chipre (EUR)
24. 0.03 - Malta (EUR)
25. 0.09 - Lituania (EUR)
26. 0.09 - Croacia (EUR)
27. 0.07 - Eslovenia (EUR)

Contenido inicial de la lista de países con mayores economías:
1. 9.90 - Estados Unidos (AME)
2. 8.50 - China (ASI)
3. 4.70 - Alemania (EUR)
4. 4.28 - Japón (ASI)
5. 4.13 - India (ASI)
6. 3.96 - Reino Unido (EUR)
7. 3.20 - Francia (EUR)
8. 2.40 - Italia (EUR)
9. 2.54 - Rusia (EUR)
10. 2.28 - Canadá (AME)
11. 2.26 - Brasil (AME)
12. 1.70 - España (EUR)
13. 1.86 - México (AME)
14. 1.86 - Corea del Sur (ASI)
15. 1.83 - Australia (OCE)
16. 1.57 - Turquía (EUR)
17. 1.44 - Indonesia (ASI)
18. 1.20 - Países Bajos (EUR)
19. 1.27 - Arabia Saudita (ASI)
20. 1.04 - Polonia (EUR)

Ordenación de la lista de países con mayores economías por nombre (alfabético):
1. 4.70 - Alemania (EUR)
2. 1.27 - Arabia Saudita (ASI)
3. 1.83 - Australia (OCE)
4. 2.26 - Brasil (AME)
5. 2.28 - Canadá (AME)
6. 8.50 - China (ASI)
7. 1.86 - Corea del Sur (ASI)
8. 1.70 - España (EUR)
9. 9.90 - Estados Unidos (AME)
10. 3.20 - Francia (EUR)
11. 4.13 - India (ASI)
12. 1.44 - Indonesia (ASI)
13. 2.40 - Italia (EUR)
14. 4.28 - Japón (ASI)
15. 1.86 - México (AME)
16. 1.20 - Países Bajos (EUR)
17. 1.04 - Polonia (EUR)
18. 3.96 - Reino Unido (EUR)
19. 2.54 - Rusia (EUR)
20. 1.57 - Turquía (EUR)

Ordenación de la lista de paises con mayores economías por PIB:
1. 1.04 - Polonia (EUR)
2. 1.20 - Países Bajos (EUR)

3. 1.27 - Arabia Saudita (ASI)
4. 1.44 - Indonesia (ASI)
5. 1.57 - Turquía (EUR)
6. 1.70 - España (EUR)
7. 1.83 - Australia (OCE)
8. 1.86 - Corea del Sur (ASI)
9. 1.86 - México (AME)
10. 2.26 - Brasil (AME)
11. 2.28 - Canadá (AME)
12. 2.40 - Italia (EUR)
13. 2.54 - Rusia (EUR)
14. 3.20 - Francia (EUR)
15. 3.96 - Reino Unido (EUR)
16. 4.13 - India (ASI)
17. 4.28 - Japón (ASI)
18. 4.70 - Alemania (EUR)
19. 8.50 - China (ASI)
20. 9.90 - Estados Unidos (AME)

Contenido final de la lista de Paises de la UE:
1. $$4.70 - Alemania (EUR)$$
2. $$3.20 - Francia (EUR)$$
3. $$2.40 - Italia (EUR)$$
4. $$1.70 - España (EUR)$$
5. $$1.20 - Países Bajos (EUR)$$
6. 0.86 - Polonia (EUR)
7. 0.66 - Bélgica (EUR)
8. 0.61 - Suecia (EUR)
9. 0.56 - Irlanda (EUR)
10. 0.54 - Austria (EUR)
11. 0.41 - Dinamarca (EUR)
12. 0.38 - Rumanía (EUR)
13. 0.34 - República Checa (EUR)
14. 0.31 - Finlandia (EUR)
15. 0.30 - Portugal (EUR)
16. 0.25 - Grecia (EUR)
17. 0.23 - Hungría (EUR)
18. 0.14 - Eslovaquia (EUR)
19. 0.11 - Bulgaria (EUR)
20. 0.09 - Luxemburgo (EUR)
21. 0.05 - Estonia (EUR)
22. 0.05 - Letonia (EUR)
23. 0.04 - Chipre (EUR)
24. 0.03 - Malta (EUR)
25. 0.09 - Lituania (EUR)
26. 0.09 - Croacia (EUR)
27. 0.07 - Eslovenia (EUR)

Contenido final de la lista de países de la UE con PIB superior a 1 Billón:
1. 4.70 - Alemania (EUR)
2. 3.20 - Francia (EUR)
3. 2.40 - Italia (EUR)
4. 1.70 - España (EUR)
5. 1.20 - Países Bajos (EUR)

Contenido final del conjunto de países de la UE con PIB superior a 1 Billón:
1. 1.70 - España (EUR)
2. 3.20 - Francia (EUR)

3. 2.40 - Italia (EUR)
4. 1.20 - Países Bajos (EUR)
5. 4.70 -

1.3.- Ejercicio 3: mapa para almacenar países y PIB por continente

Haciendo uso del array `ARRAY_PAISES_MUNDO` que tenéis declarado en la clase `Utilidades`, vamos a clasificar los países de las mayores economías mundiales por continente.

Engin_Akyurt (Pixabay License)

El formato que sigue cada uno de los objetos de tipo `String` contenidos en el array es:

- **"1.70 - España (EUR)"**, donde:
  - **"1.70"**: se corresponde con el PIB del país expresado en billones de euros. Esta cantidad siempre tendrá el mismo formato (un número entero y dos números decimales);
  - **' - '**: se corresponde con la cadena separadora entre el PIB y el nombre del país;
  - **"España"**: se corresponde con el nombre del país;
  - **" "**: se corresponde con el carácter espacio y separa el nombre del país de la abreviatura del continente;
  - **"(EUR)"**: se corresponde con el continente en formato abreviado. Siempre tendrá el formato de 3 letras encerradas entre paréntesis.

Para ello, escribiremos un programa en Java que:

1. **Declare una estructura** de tipo `Map` donde las claves serán **el continente al que pertenece cada país** (de tipo `String`) y los valores serán una lista con los países correspondientes a cada continente (también de tipo `String`).
2. **Declare otra estructura de tipo** `Map` donde las claves serán el **continente al que pertenece cada país** (de tipo String) y los valores serán la suma de los PIB de los países correspondientes a cada continente (el valor será de tipo `Double`).
3. **Convierta el array** `ARRAY_PAISES_MUNDO` **en una lista** y, mediante un bucle, la recorra extrayendo de cada posición el continente y el nombre del país y su PIB. Debemos tener cuidado con el tipado de datos, ya que el array contiene elementos de tipo `String`.
4. **Inserte en cada mapa**, los datos extraídos de forma que que el **primer mapa** contendrá al final, para cada continente la lista de países que pertenecen a ese continente y para el **segundo mapa,** se obtendrá para cada continente un único valor correspondiente a la suma de todos los PIB de los países de ese continente**.**
5. Por último, **recorra las estructuras** `Map` **obteniendo las claves y los valores, y mostrando por pantalla su contenido**.

Ejemplo de ejecución

Aquí te mostramos el resultado de la ejecución del programa. Como no hay entrada de datos, tu programa debe generar la misma salida.

```
RELACIÓN DE PAISES y ACUMULADO DE PIB ORGANIZADOS POR CONTINENTES
-----------------------------------------------------------------
Contenido del mapa de países organizados por continentes:

Países de AME: [Estados Unidos, Canadá, Brasil, México]
Países de ASI: [China, Japón, India, Corea del Sur, Indonesia, Arabia Saudita]
Países de EUR: [Alemania, Reino Unido, Francia, Italia, Rusia, España, Turquía, P
Países de OCE: [Australia]

Contenido del mapa de PIB total por continente:

Continente: AME ---> PIB Total: 16,30
Continente: ASI ---> PIB Total: 21,48
Continente: EUR ---> PIB Total: 22,31
Continente: OCE ---> PIB Total: 1,83
```

Recomendación

Para convertir un **`String`** a **`double`** puedes utilizar los métodos de la clase envoltorio **`Ooubl`**e.

2.- Información de interés

Recursos necesarios y recomendaciones

Recursos:

mohamed Hassan (Licencia Pixabay)

- Se proporciona un **proyecto base** sobre el que debes trabajar: ProyectoBase. En el que tendrás incluidas una clase (programa) para cada ejercicio (cada una con su método `main`), así como un conjunto de utilidades auxiliares en la clase `Utilidades`. **Es obligatorio utilizar este proyecto base para resolver la tarea.**

Recomendaciones:

- **Indenta** tu código de forma adecuada. Recuerda que NetBeans puede hacer esto por ti (selecciona el código a "*formatear*" o "*embellecer*" y pulsa `Alt+Mayús.+F`).
- Procura respetar las convenciones de **nombrado de variables** indicadas en la unidad 1. Recuerda que en NetBeans puedes renombrar rápidamente una variable, un método o una clase, situando el cursor encima del nombre y pulsando `Ctrl+R`.

Indicaciones de entrega

Una vez completados todos los ejercicios de la tarea, el envío se realizará a través de la plataforma. Comprime la carpeta del proyecto NetBeans en un fichero .zip y nómbralo siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_PROG07_Tarea

Recuerda que **tu proyecto NetBeans también debe llamarse así** (tanto la carpeta donde se encuentra el proyecto como el nombre del proyecto en Netbeans tienes que renombrarlos respecto al nombre original del proyecto base que vas a utilizar). **Si no lo haces, todos los proyectos se llamarán igual y la corrección de tu tarea podría confundirse con la de otra persona**.

Por tanto, **si no entregas tu tarea siguiendo estas reglas, se te devolverá para que lo hagas correctamente** .

3.- Evaluación de la tarea

Criterios de evaluación implicados

Del RA6 (*Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos*):

- b) Se han reconocido las librerías de clases relacionadas con tipos de datos avanzados.
- c) Se han reconocido las características y ventajas de cada una de la colecciones de datos disponibles.
- d) Se han utilizado listas para almacenar y procesar información.
- e) Se han utilizado iteradores para recorrer los elementos de las listas.
- f) Se han utilizado operaciones agregadas para el manejo de información almacenada en colecciones.
- g) Se han creado clases y métodos genéricos.

Peggy_Marco (Pixabay License)

¿Cómo valoramos y puntuamos tu tarea?

Cada uno de los CE cubiertos por esta tarea será evaluado a través de los criterios de corrección y puntuación indicados detalladamente en la **rúbrica** de la tarea que puedes consultar a continuación:

ÍTEMS	PUNTUACIONES			
(RA6.b,RA6.c) EJERCICIO 1: Se declaran los conjuntos necesarios de manera apropiada y con nombres siguiendo el convenio.	No se realiza correctamente. **0 puntos**	Se declaran los conjuntos con algún error. **0.15 puntos**	Se declaran los conjuntos correctamente. **0.25 puntos**	
(RA6.b,RA6.c) EJERCICIO 1: Se instancian y se rellenan los conjuntos iniciales que contienen los países. Utilización de los métodos Utilidades.getArrayPaisesUE() y Utilidades.getArrayPaisesMundo().	No se realiza correctamente. **0 puntos**	Se instancian y se rellenan los conjuntos iniciales sin usar los métodos de la clase Utilidades. **0.15 puntos**	Se instancian y se rellenan los conjuntos iniciales con algún error. **0.35 puntos**	Se instancian y se rellenan los conjuntos iniciales correctamente **0.5 puntos**
(RA6.b,RA6.c,RA6.f) EJERCICIO 1: Se instancia y se rellena el conjunto con la unión de los conjuntos iniciales haciendo uso del constructor apropiado.	No se realiza correctamente. **0 puntos**	Se instancia y se rellena el conjunto con algún error. **0.25 puntos**	Se instancia y se rellena el conjunto correctamente. **0.5 puntos**	
(RA6.b,RA6.c,RA6.f) EJERCICIO 1: Se instancia y se rellena el conjunto con la intersección de los conjuntos iniciales haciendo uso del constructor apropiado.	No se realiza correctamente. **0 puntos**	Se instancia y se rellena el conjunto con algún error. **0.25 puntos**	Se instancia y se rellena el conjunto correctamente. **0.5 puntos**	
(RA6.b,RA6.c,RA6.f) EJERCICIO 1: Se instancia y se rellena el conjunto con la diferencia de los conjuntos iniciales haciendo uso del constructor apropiado.	No se realiza correctamente. **0 puntos**	Se instancia y se rellena el conjunto con algún error. **0.25 puntos**	Se instancia y se rellena el conjunto correctamente. **0.5 puntos**	
(RA6.b,RA6.c) EJERCICIO 1: Se muestra el contenido de todos los conjuntos. Utilización del método Utilidades.mostrarPaises().	No se realiza correctamente. **0 puntos**		Se muestra el contenido de todos los conjuntos correctamente usando el método de la clase Utilidades. **0.25 puntos**	
(RA6.b,RA6.c) EJERCICIO 2: Se declaran las colecciones necesarias de manera apropiada y con nombres significativos.	No se realiza correctamente. **0 puntos**	Se declaran las colecciones con algún error. **0.15 puntos**	Se declaran las colecciones correctamente. **0.25 puntos**	
(RA6.b,RA6.c,RA6.d) EJERCICIO 2: Se instancian y se rellenan las dos listas iniciales que contienen los paises correspondientes. Utilización de los métodos Utilidades.getArrayPaisesUE() y Utilidades.getArrayPaisesMundo().	No se realiza correctamente. **0 puntos**	Se instancian y se rellenan las listas iniciales con algún error o sin usar los métodos de la clase Utilidades. **0.25 puntos**	Se instancian y se rellenan las listas iniciales correctamente, pero sin utilizar los métodos disponibles en la interfaz. **0.5 puntos**	Se instancian y se rellenan las listas iniciales correctamente utilizando los métodos disponibles en la interfaz. **0.75 puntos**

(RA6.b,RA6.c,RA6.d) EJERCICIO 2: Se instancia la lista y el conjunto para almacenar los países de la UE con PIB mayor a 1 billón de euros.	No se realiza correctamente. **0 puntos**	Se instancia la lista y el conjunto con algún error. **0.15 puntos**	Se instancia la lista y el conjunt correctamente. **0.25 puntos**	
(RA6.b,RA6.d,RA6.e) EJERCICIO 2: Se recorre la lista de países de la UE haciendo uso de iteradores.	No se realiza correctamente. **0 puntos**	Se recorre la lista de países de la UE usando iteradores con algún error. **0.25 puntos**	Se recorre la lista de países de la UE usando iteradores correctamente. **0.5 puntos**	
(RA6.b,RA6.d,RA6.e) EJERCICIO 2: Se rellena la lista y el conjunto que contiene los países de la UE con PIB mayor a 1 billón de euros, de manera correcta.	No se realiza correctamente. **0 puntos**	Se rellena la lista y el conjunto con algún error. **0.25 puntos**	Se rellena la lista y el conjunto correctamente. **0.5 puntos**	
(RA6.f) EJERCICIO 2: Se implementa correctamente la clase ComparadorPaisesPorNombre.	No se realiza correctamente. **0 puntos**	Se implementa la clase con errores. **0.15 puntos**	Se implementa la clase sin implementar la interfaz *Comparator*. **0.35 puntos**	Se implementa la clas correctamente. **0.5 puntos**
(RA6.f) EJERCICIO 2: Se implementa correctamente la clase ComparadorPaisesPorPIB.	No se realiza correctamente. **0 puntos**	Se implementa la clase con errores. **0.15 puntos**	Se implementa la clase sin implementar la interfaz *Comparator*. **0.35 puntos**	Se implementa la clas correctamente. **0.5 puntos**
(RA6.b,RA6.c) EJERCICIO 2: Se muestra el contenido de todas las listas. Utilización del método *Utilidades.mostrarPaises()*.	No se realiza correctamente. **0 puntos**		Se muestra el contenido de todas las listas correctamente usando el método de la clase Utilidades. **0.25 puntos**	
(RA6.b,RA6.c) EJERCICIO 3: Se declaran e instancian las estructuras de tipo Map de forma correcta y apropiada.	No se realiza correctamente. **0 puntos**	Se declaran e instancian las estructuras de tipo Map con algún error. **0.25 puntos**	Se declaran e instancian las estructuras de tipo Map correctamente. **0.5 puntos**	
(RA6.b,RA6.c,RA6.d) EJERCICIO 3: Se instancia y se rellena la lista inicial que contiene los países con las mayores economías mundiales. Utilización del método Utilidades.getArrayPaisesMundo().	No se realiza correctamente. **0 puntos**	Se instancia y se rellena la lista inicial con algún error o sin usar el método de la clase Utilidades. **0.15 puntos**	Se instancia y se rellena la lista inicial correctamente, pero sin utilizar los métodos disponibles en la interfaz. **0.35 puntos**	Se instancia y se rellena la lista inicial correctamente utilizando los métodos disponibles en la interfaz. **0.5 puntos**
(RA6.b,RA6.d) EJERCICIO 3: Se recorre la lista inicial de forma correcta y apropiada.	No se realiza correctamente. **0 puntos**	Se recorre la lista inicial con algún error. **0.25 puntos**	Se recorre la lista inicial correctamente. **0.5 puntos**	
(RA6.b,RA6.d) EJERCICIO 3: Se rellena el mapa de países por continente de manera correcta con las coincidencias encontradas.	No se realiza correctamente. **0 puntos**	Se rellena el mapa con muchos errores **0.15 puntos**	Se rellena el mapa con errores en el procesamiento del mapa. **0.3 puntos**	Se rellena el mapa con errores en el procesamiento de la cadena. **0.5 puntos**	Se rellena el map correctamente. **0.75 puntos**
(RA6.b,RA6.d) EJERCICIO 3: Se rellena el mapa de PIB acumulado por continente de manera correcta con las coincidencias encontradas.	No se realiza correctamente. **0 puntos**	Se rellena el mapa con muchos errores **0.15 puntos**	Se rellena el mapa con errores en el procesamiento del mapa. **0.3 puntos**	Se rellena el mapa con errores en el procesamiento de la cadena. **0.5 puntos**	Se rellena el mapa correctamente. **0.75 puntos**
(RA6.b,RA6.d,RA6.e) EJERCICIO 3: Se recorre el mapa para mostrarlo con el formato apropiado: una línea por cada par (clave, valor).	No se realiza correctamente. **0 puntos**	Se muestra el mapa con errores en el procesamiento del mapa. **0.35 puntos**	Se muestra el mapa con errores en el formato de salida. **0.7 puntos**	Se muestra el mapa correctamente. **1 punto**

Aquí tienes algunas cuestiones generales que podrán penalizarte en la evaluación de los ejercicios de la tarea. Tenlos muy en cuenta al resolver los ejercicios y comprobar su funcionamiento:

- **No se utilizan** en los casos que sea posible, **los métodos disponibles en la clase `Utilidades` para optimizar el código y evitar realizar operaciones innecesarias. En estos casos la penalización podrá ser importante, incluso llegándose a poder puntuar con 0 ese ejercicio.**
- **Cada uno de los programas debe compilar.** Cualquier funcionalidad pedida en el enunciado debe poderse probar y ha de funcionar correctamente de acuerdo a las especificaciones del enunciado. **Si la clase o el programa que la prueba ni siquiera compila, el ejercicio podría considerarse nulo y su puntuación podría llegar a ser 0.**

- **La ejecución de los programas no debe romperse**. Si eso sucede es porque la clase no ha sido implementada correctamente con respecto a los requisitos indicados en el enunciado.
- **Se utilizan estructuras de salto incondicional para el control del flujo**, o bien herramientas como `return`, `System.exit()` o cualquier otro mecanismo que no sea el fin de la ejecución natural del programa.
- **Se declaran identificadores que no cumplen con el convenio sobre "asignación de nombres a identificadores"** establecido para el lenguaje Java para constantes, variables, métodos, clases, etc.
- **Se declaran identificadores que no tienen nombres significativos o descriptivos** que representen de alguna manera la información que están almacenando para que el código quede lo más claro, legible y autodocumentado posible.
- **No se observa una corrección ortográfica y gramatical,** así como la coherencia en las expresiones lingüísticas, tanto en los comentarios en el código como en los textos de los mensajes que aparezcan en pantalla para pedir información de entrada al usuario, mostrar resultados de salida o indicar situaciones de error. Deben evitarse mensajes de entrada de datos, salida de resultados inapropiados y/o error, descontextualizados, insuficientes o incorrectos.
- **El código no está correctamente indentado**. Es fundamental para poder observar e intuir rápidamente, y de forma visual, la estructura de los programas. Recuerda que NetBeans puede hacer esto por ti (selecciona el código a "formatear" o "embellecer" y **pulsa Alt+Mayús.+F**).

Se valorará especialmente en cada uno de los apartados:

- Que la **respuesta** sea **personal** y no copiada de Internet o de otras fuentes.
- Que sea **coherente** con el resto de respuestas y con los conceptos de la unidad.
- Que muestre que se **comprenden** adecuadamente los **conceptos** tratados.

IMPORTANTE: Motivos de devolución de la tarea o de calificación mínima

La tarea obtendrá automáticamente una calificación de **cero** si:

- No se entrega **un único proyecto** que incluya los tres ejercicios que se piden en la tarea utilizando el IDE NetBeans.
- Alguno de los programas presenta **errores de compilación** impidiendo su ejecución y prueba.
- **No se respeta el proyecto base** que se entrega al alumno para realizar la tarea.
- Se utilizan **contenidos** en alguno de los ejercicios del proyecto **que no se han estudiado** en la unidad actual o anteriores a la misma.
- Se presenta un proyecto total o parcialmente copiado de otro alumno.

Recordatorio: Si una tarea obtiene calificación cero por alguno de los motivos previamente mencionados, será devuelta al alumno. Este tendrá derecho a una segunda entrega únicamente en los siguientes casos:

- Que se trate de la primera vez que presenta la tarea.
- Que la fecha de entrega inicial haya sido al menos una semana antes de la fecha límite establecida para dicha tarea.