

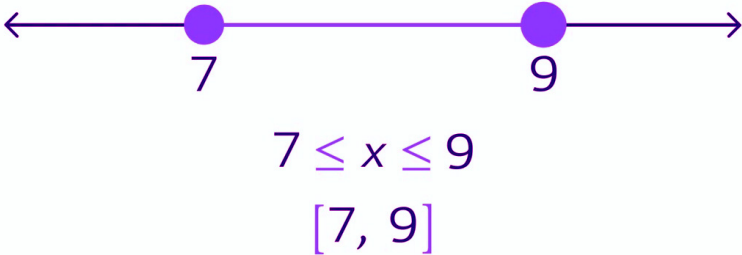
UT5
José R. Mas Davó authored 3 weeks ago

85519756

M+ UT5Problema2.md 6.08 KiB

UT5 - Problema 2: Gestió d'interval·ls per a una aplicació de planificació

Agrupament: Individual



Pregunta guia

Com podem implementar una classe `Interval` que permeti representar, manipular i analitzar rangs numèrics per a una aplicació de planificació i anàlisi de dades?

Objectius d'aprenentatge

En completar aquesta pràctica, sereu capaços de:

- Aplicar conceptes de programació orientada a objectes com encapsulació i sobrecàrrega de constructors.
- Implementar operacions de comparació i modificació d'objectes.
- Reconèixer la sintaxi, estructura i components típics d'una classe en Java.
- Definir classes, atributs i mètodes.
- Implementar constructors per inicialitzar objectes.
- Desenvolupar programes que instancien i utilitzen objectes de classes definides.
- Utilitzar mecanismes per controlar la visibilitat dels membres de la classe.

Context i descripció del problema

Una empresa de desenvolupament de software està creant una aplicació de planificació i anàlisi de dades que necessita gestionar interval·ls numèrics de manera eficient. Aquestes funcionalitats seran utilitzades en diversos mòduls de l'aplicació, com ara planificació de projectes, anàlisi de temps i gestió de recursos. Us han contractat per completar la implementació de la classe `Interval`, que serà clau per a aquestes funcionalitats.

La classe `Interval` ja té definides les capçaleres dels mètodes necessaris i la documentació corresponent per a cada mètode, però encara no s'ha implementat. La tasca consisteix a desenvolupar aquests mètodes per proporcionar les funcionalitats descrites a la documentació de la classe.

Detalls de la classe `Interval`

Atributs

- `inferior`: un `double` que representa el límit inferior de l'interval.
- `superior`: un `double` que representa el límit superior de l'interval.

Constructors

1. `public Interval(double inferior, double superior)`: Crea un interval amb els límits proporcionats.
2. `public Interval(double superior)`: Crea un interval amb el límit superior proporcionat i un límit inferior per defecte.
3. `public Interval(Interval interval)`: Constructor còpia, crea un interval a partir d'un altre.
4. `public Interval()`: Constructor per defecte, crea un interval amb valors per defecte.

Mètodes d'instància

- `public Interval clone()` : Clona l'interval actual.
- `public double longitud()` : Retorna la longitud de l'interval.
- `public void moure(double moviment)` : Mou els límits de l'interval segons el moviment proporcionat.
- `public Interval mogut(double moviment)` : Retorna un nou interval mogut segons el moviment proporcionat.
- `public boolean inclou(double valor)` : Determina si un valor està dins de l'interval.
- `public boolean inclou(Interval interval)` : Determina si un interval està inclòs dins de l'interval actual.
- `public boolean equals(Interval interval)` : Compara dos intervals per veure si són iguals.
- `public Interval interseccio(Interval interval)` : Retorna l'interval que resulta de la intersecció amb un altre interval.
- `public boolean intersecta(Interval interval)` : Determina si dos intervals s'intersecten.
- `public void oposar()` : Inverteix els límits de l'interval.
- `public void doblar()` : Doble la longitud de l'interval.
- `public void recollir()` : Demana a l'usuari que introduïska els límits de l'interval.
- `public void mostrar()` : Mostra l'interval amb el format `[limitInferior, limitSuperior]`.
- `public Interval[] trossejar(int trossos)` : Retorna un array d'intervals que divideixen l'interval en parts iguals.

Instruccions

1. Investigació preliminar:

- Repassa els conceptes de la Programació Orientada a Objectes (POO), incloent-hi classes, atributs, mètodes i constructors.
- Fes una revisió dels conceptes matemàtics d'intervals numèrics i operacions sobre ells, com ara interseccions i longituds.

2. Desenvolupament del programa:

- Implementa la classe Interval seguint les capçaleres de mètodes i la documentació proporcionada.
- Assegura't que cada mètode compleix amb la seva funcionalitat descrita.
- **Important:** REUTILITZA CODI (utilitza mètodes ja creats). És un concepte fonamental de la POO.

3. Proves i depuració:

- Crea una classe de prova (ProvaInterval) per verificar el correcte funcionament de tots els mètodes.
- Prova el programa amb diversos intervals per garantir que totes les funcionalitats es comporten correctament.
- Depura qualsevol error o comportament inesperat.

4. Reflexió i documentació:

- Afegeix comentaris addicionals si és necessari per explicar la lògica complexa.
- Prepara un breu informe que incloga:
 - Descripció de qualsevol desafiament que hages enfrontat durant la implementació i com l'has superat.
 - Suggeriments per a possibles millores o extensions de la classe Interval.

Exemple de funcionament del programa

```
Interval i1 = new Interval(1, 5);
Interval i2 = new Interval(3, 7);

System.out.println("Longitud de i1: " + i1.longitud()); // Hauria de mostrar: 4.0
System.out.println("i1 inclou 3? " + i1.inclou(3)); // Hauria de mostrar: true
System.out.println("i1 intersecta amb i2? " + i1.intersecta(i2)); // Hauria de mostrar: true

Interval interseccio = i1.interseccio(i2);
interseccio.mostrar(); // Hauria de mostrar: [3.0, 5.0]

i1.doblar();
i1.mostrar(); // Hauria de mostrar: [-1.0, 7.0]

Interval[] trossos = i1.trossejar(4);
for (Interval tros : trossos) {
    tros.mostrar();
}
// Hauria de mostrar:
// [-1.0, 1.0]
// [1.0, 3.0]
// [3.0, 5.0]
// [5.0, 7.0]
```

Entrega

- Envia el fitxer del projecte exportat de NetBeans (`UT5Problema2.zip`) i l'informe en format PDF.
- Recorda seguir la convenció de noms.
- **Important:** Afig el teu nom i cognoms sempre com a comentari al principi dels fitxers JAVA (En NetBeans després de `@author`).

Extensió opcional

- Implementa un mètode que permeti comparar dos intervals per veure si són més grans, menors o iguals en longitud.
- Implementa un mètode per unir dos intervals si es superposen.