Commencé le	vendredi 3 février 2023, 09:33
État	Terminé
Terminé le	vendredi 3 février 2023, 11:51
Temps mis	2 heures 17 min
Points	7,00/9,00
Note	**7,78** sur 10,00 (**77,78**%)

Description

Le but de ce TD noté est d'implémenter un micro moteur de résolution de contraintes.

You must stay on the TD6 test page, without using any IDE, application or other web page.

Some tests may be hidden. Other tests could be added after the exam.

In case of contradiction between statements and tests, only consider the tests.

Each question is independant.

Question 1

Correct

Note de 1,00 sur 1,00

Write a VectorOpt class that uses a vector as backend to provide a vector of arbitrary objects where elements can be absent. The class must provide the following methods : a constructor with a size, get, set, stream (stream of the present values), peek (a selection of the vector entries), and complete (true if all values are present)

> Reminder: you can use wildcards <?> to represent an arbitrary generic type (e.g. Set<?>)

Par exemple:

Test	Résultat

`VectorOpt vectorOpt = new VectorOpt(5);`
`vectorOpt.set(0, 10);`
`vectorOpt.set(2, "A");`
`System.out.println(vectorOpt.get(0));`
`System.out.println(vectorOpt.get(1));`
`System.out.println(vectorOpt.get(2));`
	`Optional[10]`
	`Optional.empty`
	`Optional[A]`

`VectorOpt vectorOpt = new VectorOpt(5);`
`vectorOpt.set(0, 10);`
`vectorOpt.set(2, 30);`
`Stream<?> s = vectorOpt.stream();`
`System.out.println("Sum of present values: "`
`  + s.mapToInt(x -> ((Integer) x).intValue()).sum());`
	`Sum of present values: 40`

`VectorOpt vectorOpt = new VectorOpt(5);`
`vectorOpt.set(0, 10);`
`vectorOpt.set(2, 20);`
`vectorOpt.set(3, "C");`
`VectorOpt sub = vectorOpt.peek(List.of(0,3,4)); // Entries 0 3 and 4`
`System.out.println("0: " + sub.get(0));`
`System.out.println("1: " + sub.get(1));`
`System.out.println("2: " + sub.get(2));`
	`0: Optional[10]`
	`1: Optional[C]`
	`2: Optional.empty`

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
public class VectorOpt{
    private List<Optional<?>> vec = new ArrayList<>();

    // a constructor with a size
    public VectorOpt(int size){
        for (int i = 0; i<size; i++){
            vec.add(Optional.empty());
        }
    }

    // get
    public Object get(int index) {
        if (vec.get(index).isEmpty())
            return Optional.empty();
        return vec.get(index);
    }

    // set
```

	Test	Résultat attendu	Résultat obtenu	

✓	`VectorOpt vectorOpt = new VectorOpt(5);`
	`vectorOpt.set(0, 10);`
	`vectorOpt.set(2, "A");`
	`System.out.println(vectorOpt.get(0));`
	`System.out.println(vectorOpt.get(1));`
	`System.out.println(vectorOpt.get(2));`
		`Optional[10]`
		`Optional.empty`
		`Optional[A]`
			`Optional[10]`
			`Optional.empty`
			`Optional[A]`
				✓

✓	`VectorOpt vectorOpt = new VectorOpt(5);`
	`vectorOpt.set(0, 10);`
	`vectorOpt.set(2, 30);`
	`Stream<?> s = vectorOpt.stream();`
	`System.out.println("Sum of present values: "`
	`  + s.mapToInt(x -> ((Integer) x).intValue()).sum());`
		`Sum of present values:`
		`40`
			`Sum of present values:`
			`40`
				✓

✓	`VectorOpt vectorOpt = new VectorOpt(5);`
	`vectorOpt.set(0, 10);`
	`vectorOpt.set(2, 20);`
	`vectorOpt.set(3, "C");`
	`VectorOpt sub = vectorOpt.peek(List.of(0,3,4)); // Entries 0 3 and 4`
	`System.out.println("0: " + sub.get(0));`
	`System.out.println("1: " + sub.get(1));`
	`System.out.println("2: " + sub.get(2));`
		`0: Optional[10]`
		`1: Optional[C]`
		`2: Optional.empty`
			`0: Optional[10]`
			`1: Optional[C]`
			`2: Optional.empty`
				✓

✓	`VectorOpt vectorOpt = new VectorOpt(3);`
	`vectorOpt.set(0, 10);`
	`vectorOpt.set(1, 20);`
	`System.out.println(vectorOpt.complete());`
	`vectorOpt.set(2, 30);`
	`System.out.println(vectorOpt.complete());`
		`false`
		`true`
			`false`
			`true`
				✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 1,00/1,00.

Description

Dans les exercices qui suivent la classe VectorOpt est fournie, ainsi que l'interface Constraint qui est un prédicat sur un vecteur de valeurs.

```
public interface Constraint extends Predicate<VectorOpt> {
}
```

Question 2

Correct

Note de 1,00 sur 1,00

Write a static method allDifferent that returns a constraint that enforces that all values are different (empty values are not taken into account).

Par exemple:

Test	Résultat

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`System.out.println(allDifferent().test(v))`
	`true`

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`v.set(3, 10);`
`System.out.println(allDifferent().test(v))`
	`false`

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
public static Constraint allDifferent() {
    return v -> v.stream().distinct().count() == v.stream().count();
}
```

	Test	Résultat attendu	Résultat obtenu	

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`System.out.println(allDifferent().test(v))`
		`true`	`true`	✓

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`v.set(3, 10);`
	`System.out.println(allDifferent().test(v))`
		`false`	`false`	✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 1,00/1,00.

Question 3

Correct

Note de 1,00 sur 1,00

Write a static method allEqual that returns a constraint that enforces that all values are equals (empty values are not taken into account).

Par exemple:

Test	Résultat

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`System.out.println(allEqual().test(v))`
	`false`

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 10);`
`v.set(3, 10);`
`System.out.println(allEqual().test(v))`
	`true`

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
public static Constraint allEqual() {
    return v -> v.stream().distinct().count() == 1;
}
```

	Test	Résultat attendu	Résultat obtenu	

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`System.out.println(allEqual().test(v))`
		`false`	`false`	✓

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 10);`
	`v.set(3, 10);`
	`System.out.println(allEqual().test(v))`
		`true`	`true`	✓

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, "A");`
	`v.set(2, "A");`
	`System.out.println(allEqual().test(v))`
		`true`	`true`	✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 1,00/1,00.

Question 4

Correct

Note de 1,00 sur 1,00

Write a static method exactSum that returns a constraint enforcing that the sum of the given variables are equal to a given number.

Par exemple:

Test	Résultat

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`System.out.println(exactSum(List.of(2),30).test(v));`
`System.out.println(exactSum(List.of(2),20).test(v));`
	`true`
	`false`

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`System.out.println(exactSum(List.of(0,2),30).test(v));`
`System.out.println(exactSum(List.of(0,2),40).test(v));`
	`false`
	`true`

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
public static Constraint exactSum(List<Integer> indexes, int goal) {
    return v -> {
        var peek = v.peek(indexes);
        return peek.stream().mapToInt(x -> (int) x).sum() == goal;
    };
}
```

	Test	Résultat attendu	Résultat obtenu	

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`System.out.println(exactSum(List.of(2),30).test(v));`
	`System.out.println(exactSum(List.of(2),20).test(v));`
		`true`
		`false`
			`true`
			`false`
				✓

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`System.out.println(exactSum(List.of(0,2),30).test(v));`
	`System.out.println(exactSum(List.of(0,2),40).test(v));`
		`false`
		`true`
			`false`
			`true`
				✓

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`System.out.println(exactSum(List.of(0,1),10).test(v));`
		`true`	`true`	✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 1,00/1,00.

Question 5

Correct

Note de 1,00 sur 1,00

Write a static method inSet that returns a constraint enforcing that all the values of given variables are present in the given set (empty values are not taken into account).

Par exemple:

Test	Résultat

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`v.set(3, "A");`
`System.out.println(inSet(List.of(2),Set.of(30)).test(v));`
`System.out.println(inSet(List.of(2),Set.of(10,20)).test(v));`
	`true`
	`false`

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`v.set(3, "A");`
`System.out.println(inSet(List.of(0,1,2,3),Set.of()).test(v));`
`System.out.println(inSet(List.of(0,1,2,3),Set.of(10,20,30,"A","B")).test(v));`
	`false`
	`true`

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
public static Constraint inSet(List<Integer> indexes, Set<Object> goal){
    return v -> {
        var peek = v.peek(indexes);
        var peekCount = peek.stream().count();
        return peek.stream().filter(goal::contains).count() == peekCount;
    };
}
```

	Test	Résultat attendu	Résultat obtenu	

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`v.set(3, "A");`
	`System.out.println(inSet(List.of(2),Set.of(30)).test(v));`
	`System.out.println(inSet(List.of(2),Set.of(10,20)).test(v));`
		`true`
		`false`
			`true`
			`false`
				✓

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`v.set(3, "A");`
	`System.out.println(inSet(List.of(0,1,2,3),Set.of()).test(v));`
	`System.out.println(inSet(List.of(0,1,2,3),Set.of(10,20,30,"A","B")).test(v));`
		`false`
		`true`
			`false`
			`true`
				✓

Tous les tests ont été réussis ! ✓

Correct

Note pour cet envoi : 1,00/1,00.

Question **6**

Correct

Note de 1,00 sur 1,00

Write a static method someInSet that returns a constraint enforcing that some of the values of given variables are present in the given set (empty values are not taken into account).

Par exemple:

Test	**Résultat**

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`v.set(3, "A");`
`System.out.println(someInSet(List.of(2),Set.of(30)).test(v));`
`System.out.println(someInSet(List.of(2),Set.of(10,20)).test(v));`
	`true`
	`false`

`VectorOpt v = new VectorOpt(5);`
`v.set(0, 10);`
`v.set(2, 30);`
`v.set(3, "A");`
`System.out.println(someInSet(List.of(0,1,2,3),Set.of()).test(v));`
`System.out.println(someInSet(List.of(0,1,2,3),Set.of(20,30)).test(v));`
	`false`
	`true`

Réponse : (régime de pénalités : 0 %)

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
public static Constraint someInSet (List<Integer> indexes, Set<Object> goal){
    return v -> {
        var peek = v.peek(indexes);
        var peekCount = peek.stream().count();
        return peek.stream().filter(goal::contains).count() > 0;
    };
}
```

	Test	**Résultat attendu**	**Résultat obtenu**	

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`v.set(3, "A");`
	`System.out.println(someInSet(List.of(2),Set.of(30)).test(v));`
	`System.out.println(someInSet(List.of(2),Set.of(10,20)).test(v));`
		`true`
		`false`
			`true`
			`false`
				✓

✓	`VectorOpt v = new VectorOpt(5);`
	`v.set(0, 10);`
	`v.set(2, 30);`
	`v.set(3, "A");`
	`System.out.println(someInSet(List.of(0,1,2,3),Set.of()).test(v));`
	`System.out.println(someInSet(List.of(0,1,2,3),Set.of(20,30)).test(v));`
		`false`
		`true`
			`false`
			`true`
				✓

Tous les tests ont été réussis ! ✓

Correct

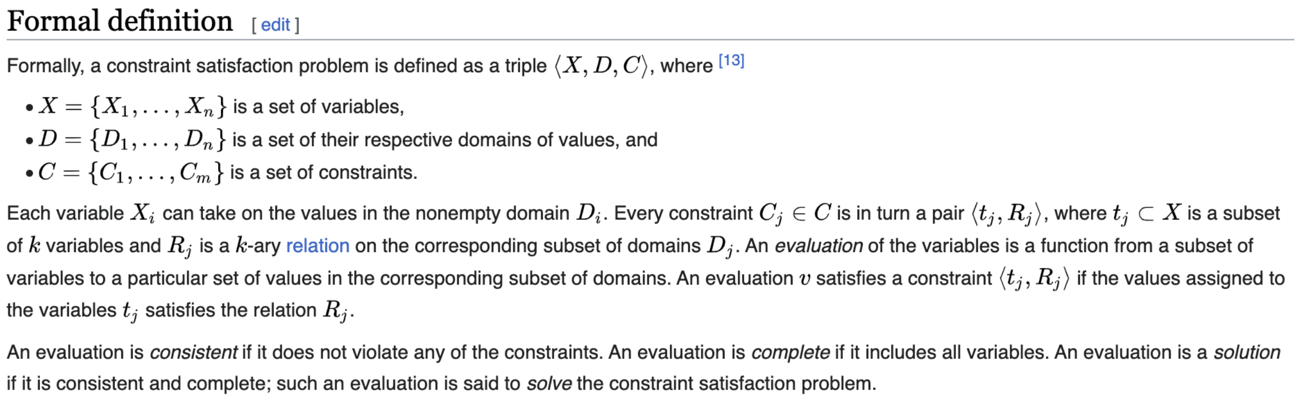
Note pour cet envoi : 1,00/1,00.

Question 7

Partiellement correct

Note de 0,67 sur 1,00

The class CSP represent a constraint satisfaction problem. Here is a formal definition of CSP from Wikipedia :



Formal definition [edit]

Formally, a constraint satisfaction problem is defined as a triple $\langle X, D, C \rangle$, where [13]

- $X = \{X_1, \ldots, X_n\}$ is a set of variables,
- $D = \{D_1, \ldots, D_n\}$ is a set of their respective domains of values, and
- $C = \{C_1, \ldots, C_m\}$ is a set of constraints.

Each variable X_i can take on the values in the nonempty domain D_i. Every constraint $C_j \in C$ is in turn a pair $\langle t_j, R_j \rangle$, where $t_j \subset X$ is a subset of k variables and R_j is a k-ary relation on the corresponding subset of domains D_j. An *evaluation* of the variables is a function from a subset of variables to a particular set of values in the corresponding subset of domains. An evaluation v satisfies a constraint $\langle t_j, R_j \rangle$ if the values assigned to the variables t_j satisfies the relation R_j.

An evaluation is *consistent* if it does not violate any of the constraints. An evaluation is *complete* if it includes all variables. An evaluation is a *solution* if it is consistent and complete; such an evaluation is said to *solve* the constraint satisfaction problem.

In our class

- Variables are represented by a List<String> (variable names)
- Domains are represented by a Map<String,List<?>> (variable names associated to possible values)
- Constraints are represented by a collection of functions: List<Constraint>

Implement the two **constraintFactory** method that takes

- variables names and
- a predicate that takes as many arguments that the variables names given

and produce the corresponding constraint applied on the correct variables. The predicate should return true if either variable is not present.

Par exemple:

Test	Résultat

`CSP pb = new CSP();`
`pb.addVariable("X", Stream.of(1, 2, 3));`
`Constraint c = pb.constraintFactory("X",`
`  x -> ((Integer) x).intValue() > 2);`
`VectorOpt v = new VectorOpt(1);`
`v.set(0, 3);`
`System.out.println(c.test(v));`
`v.set(0, 1);`
`System.out.println(c.test(v));`
	`true`
	`false`

`CSP pb = new CSP();`
`pb.addVariable("X", Stream.of(1, 2, 3));`
`pb.addVariable("Y", Stream.of(1, 2, 3));`
`Constraint c = pb.constraintFactory("X", "Y",`
`  (x, y) -> ((Integer) x).intValue() + ((Integer) y).intValue() > 2);`
`VectorOpt v = new VectorOpt(2);`
`v.set(0, 1);`
`v.set(1, 1);`
`System.out.println(c.test(v));`
`v.set(1, 3);`
`System.out.println(c.test(v));`
	`false`
	`true`

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
@SuppressWarnings("unchecked")
public class CSP  {
    private Map<String, List<?>> variables = new HashMap<>();
    private Map<String, Integer> variableNumber = new HashMap<>();
    private List<Constraint> constraints = new ArrayList<>();

    public void addVariable(String name, Stream<?> domain) {
        variables.put(name, domain.collect(Collectors.toList()));
        variableNumber.put(name, variableNumber.size());
    }

    public List<String> variables() {
        return new ArrayList<>(variables.keySet());
    }

    public void addConstraint(Constraint c) {
        constraints.add(c);
    }
```

	Test	Résultat attendu	Résultat obtenu	

✓	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(1, 2, 3));`
	`Constraint c = pb.constraintFactory("X",`
	`  x -> ((Integer) x).intValue() > 2);`
	`VectorOpt v = new VectorOpt(1);`
	`v.set(0, 3);`
	`System.out.println(c.test(v));`
	`v.set(0, 1);`
	`System.out.println(c.test(v));`
		true
		false
			true
			false
				✓

✗	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(1, 2, 3));`
	`pb.addVariable("Y", Stream.of(1, 2, 3));`
	`Constraint c = pb.constraintFactory("Y",`
	`  x -> ((Integer) x).intValue() > 2);`
	`VectorOpt v = new VectorOpt(2);`
	`v.set(1, 1);`
	`System.out.println(c.test(v));`
	`v.set(1, 3);`
	`System.out.println(c.test(v));`
		false
		true
			true
			true
				✗

✗	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(1, 2, 3));`
	`pb.addVariable("Y", Stream.of(1, 2, 3));`
	`Constraint c = pb.constraintFactory("X", "Y",`
	`  (x, y) -> ((Integer) x).intValue() + ((Integer) y).intValue() > 2);`
	`VectorOpt v = new VectorOpt(2);`
	`v.set(0, 1);`
	`v.set(1, 1);`
	`System.out.println(c.test(v));`
	`v.set(1, 3);`
	`System.out.println(c.test(v));`
		false
		true
			false
			false
				✗

✓	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(1, 2, 3));`
	`Constraint c = pb.constraintFactory("X",`
	`  x -> ((Integer) x).intValue() > 2);`
	`VectorOpt v = new VectorOpt(1);`
	`System.out.println(c.test(v));`
		true	true	✓

✗	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(1, 2, 3));`
	`pb.addVariable("Y", Stream.of(1, 2, 3));`
	`Constraint c = pb.constraintFactory("X", "Y",`
	`  (x, y) -> ((Integer) x).intValue() + ((Integer) y).intValue() > 2);`
	`VectorOpt v = new VectorOpt(2);`
	`v.set(0, 1);`
	`System.out.println(c.test(v));`
		true	false	✗

	Test	Résultat attendu	Résultat obtenu	

✓	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(1, 2, 3));`
	`pb.addVariable("Y", Stream.of(1, 2, 3));`
	`Constraint c = pb.constraintFactory("X",`
	`  x -> ((Integer) x).intValue() > 2);`
	`VectorOpt v = new VectorOpt(2);`
	`v.set(0, 1);`
	`System.out.println(c.test(v));`
		`false`	`false`	✓

Montrer les différences

Partiellement correct

Note pour cet envoi : 0,50/1,00.

Question 8

Partiellement correct

Note de 0,33 sur 1,00

In the CSP class write a solve method.

The solve method should use a recursive backtracking algorithm, where it iterates over the values in the domain of the current variable and adds them to the solution one by one and go recursively on the next variable.
When all variables have been assigned a value and all constraints are satisfied, the algorithm returns true. If a variable cannot be assigned a value that satisfies the constraints, the algorithm automatically backtracks to the previous variable and tries the next value in its domain. If all values have been tried and no solution has been found, the algorithm returns false.

> You can add other variable if needed. Think about using streams (e.g. `variables.get(vars.get(k)).stream();`)

Par exemple:

Test	Résultat

`CSP pb = new CSP();`
`pb.addVariable("X", Stream.of(1, 2, 4));`
`pb.addConstraint(pb.constraintFactory("X", x -> ((Integer) x).intValue() > 2));`
`System.out.println(pb.solve());`
	`Optional[{[Optional[4]]}]`

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
    public Optional<VectorOpt> solve() {
        var a = new VectorOpt(1);
        a.set(0, 4);
        return Optional.of(a);
    }

    private boolean backtrack(VectorOpt solution, List<String> vars, int k) {
        return false;// To fill
    }
```

	Test	Résultat attendu	Résultat obtenu	

✓	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(1, 2, 4));`
	`pb.addConstraint(pb.constraintFactory("X", x -> ((Integer) x).intValue() > 2));`
	`System.out.println(pb.solve());`
		`Optional[{[Optional[4]]}]`	`Optional[{[Optional[4]]}]`	✓

	Test	Résultat attendu	Résultat obtenu	

✖	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(1, 4, 2));`
	`pb.addVariable("Y", Stream.of("A", "BB", "CCC"));`
	`pb.addConstraint(pb.constraintFactory("X", x -> ((Integer) x).intValue() >= 2));`
	`pb.addConstraint(pb.constraintFactory("Y", x -> ((String) x).length() <= 2));`
	`System.out.println(pb.solve());`
		`Optional[{[Optional[4], Optional[A]]}]`	`Optional[{[Optional[4]]}]`	✖

✖	`CSP pb = new CSP();`
	`pb.addVariable("X", Stream.of(0, 4, 3));`
	`pb.addVariable("Y", Stream.of(2, 3, 4));`
	`pb.addConstraint(Constraint.exactSum(List.of(0,1), 6));`
	`pb.addConstraint(Constraint.allEqual());`
	`System.out.println(pb.solve());`
		`Optional[{[Optional[3], Optional[3]]}]`	`Optional[{[Optional[4]]}]`	✖

Montrer les différences

Partiellement correct

Note pour cet envoi : 0,33/1,00.

Question 9

Incorrect

Note de 0,00 sur 1,00

In the CSP class write a solver method that returns a stream of all possible solution.

You can add other methods if needed.

Par exemple:

Test	Résultat

`CSP pb = new CSP();`
`pb.addVariable("X", Stream.of(1, 5, 2, 4, 6));`
`pb.addConstraint(pb.constraintFactory("X", x -> ((Integer) x).intValue() > 2));`
`Stream<Vector> s = pb.solver();`
`System.out.println(s.collect(Collectors.toList()));`
	`[[5], [4], [6]]`

Réponse : (régime de pénalités : 10, 20, ... %)

Réinitialiser la réponse

L'éditeur Ace n'est pas prêt. Recharger peut-être la page ?
Falling back to raw text area.

```
    public Stream<VectorOpt> solver() {
        return solve().stream();
    }
```

Erreur(s) de syntaxe

```
__tester__.java:146: error: incompatible types: Stream<VectorOpt> cannot be converted to Stream<Vector>
Stream<Vector> s = pb.solver();
                            ^
1 error
```

Incorrect

Note pour cet envoi : 0,00/1,00.