B1_BP_INT01_20221013

01 On rappelle que la **sortie** d'un programme consiste ici en ce qui est affiché sur l'écran **et rien d'autre**. Si vous souhaitez fournir des explications pour clarifier votre pensée (non demandé), elles doivent être fournies à part : **la sortie doit être encadrée et complètement isolée sur votre copie**. Donnez la sortie du programme suivant :

```
int nb1 = 10;
int nb2 = 2;

System.out.println(nb1 + nb2);
System.out.println("nb1 + nb2");
System.out.println(nb1 + "," + nb2);

if (nb1 <= nb2) {
  System.out.println("ici");
}

nb2 = nb1 + nb2;

if (nb1 > nb2) {
  nb1 = 0;
  System.out.println("là");
} else {
  nb2 = 0;
  System.out.println("coucou");
}
  // Suite à droite
```

```
if (nb1 > nb2) {
  System.out.println("101");
} else if (nb1 == nb2) {
  System.out.println("102");
} else {
  System.out.println("103");
}

String s = "hello";
for (int i = 1; i <= nb1; i++) {
  System.out.println(s + i + " : " + nb2);
  nb2 = nb2 + 10;
}

int compt = 5;
boolean cont = true;
while (cont) {
  System.out.println(s);
  compt = compt - 2;
  if (compt == 0) {
    cont = false;
  }
}
```

02 Vous intervenez dans la phase de développement initiale d'une application web de type *e-commerce*. Vous développez une version prototype du code qui va permettre la pagination lors de l'affichage d'un catalogue de produits. On vous demande de compléter le code donné page suivante, censé calculer et afficher le nombre de pages nécessaires à l'affichage de tous les produits, pour ensuite procéder à l'affichage de la totalité des pages une par une, sur la sortie texte standard, selon le modèle suivant :

```
PAGE 1 :
[contenu de la page contenant les 25 premiers produits]
PAGE 2 :
[contenu de la page contenant les 25 produits suivants]
...et ainsi de suite...
```

Vous disposez d'une méthode utilitaire déjà écrite et testée, appelée `afficherPageSuivante` qui est capable d'afficher une page complète (c'est-à-dire une partie indiquée `[contenu de la page contenant…]` dans l'exemple ci-dessus). Cette méthode « retient » automatiquement au fur et à mesure à quelle page elle en est, ce qui fait que plusieurs appels différents affichent bien les pages les unes après les autres. Par exemple, le code suivant affiche les 3 premières pages du catalogue :

```
afficherPageSuivante();
afficherPageSuivante();
afficherPageSuivante();
```

Le développeur en charge de ce code attire votre attention sur le fait que cette méthode n'affiche pas la partie « `PAGE x :` » demandée par le besoin exprimé ci-dessus, et qui est donc laissé à votre charge.

Sur votre copie, fournissez les 2 parties de code, qui devront être intitulées *BLOC 1* et *BLOC 2*, et qui sont destinées à venir se placer aux endroits indiqués dans les commentaires correspondants du code.

```
public static void main(String[] args) {
  // Pour le test, on considérera qu'il y a 1000 produits dans le catalogue
  // et qu'il faut afficher 25 produits par page.
  // Cependant, faites en sorte que ces valeurs puissent être facilement modifiées
  // par la suite lorsqu'elles proviendront d'une base de données

  // ---------------------------------------------------------------------
  //  Calcul du nombre de pages total - À COMPLÉTER

  // ---------------------------------------------------------------------

  // Affichage du nombre de pages total
  System.out.println("Il faut " + nbPages + " pages pour afficher tous les produits");

  // ---------------------------------------------------------------------
  //  Affichage de toutes les pages selon le modèle spécifié - À COMPLÉTER

  // ---------------------------------------------------------------------
}
```

03 Considérez la version du Jeu de l'Oie ci-dessous, dans laquelle on avait cinq lancers de dé pour la partie. On veut mettre en place les modifications suivantes :

- Lorsqu'on dépasse la case gagnante (ici 20), on recule d'autant de cases que le lancer de dé dépasse ; par exemple, si on est à la case 18 et qu'on fait un 5, on se retrouve non pas à la case 23 mais à la case 17 (on va jusque 20 et on recule).
- Dans cet exemple, le calcul de la case d'arrivée 17 s'obtient par l'opération `40 - 23` ; ce calcul fonctionne, quand on dépasse 20, pour n'importe quelle case d'arrivée (ici 23).
- Le jeu doit continuer indéfiniment tant que le pion n'arrive pas pile sur la case 20 (par exemple, en étant sur la case 17, il nous faut absolument un 3 pour gagner) ; il n'y a donc plus de partie perdue.
- À la fin, on doit afficher le nombre de lancers qui ont été nécessaires pour gagner.

Les questions 03.1, 03.2 et 03.3 ne nécessitent pas la présentation du code précis, seulement une explication en une ou deux phrases.

03.1 On ne connaît donc plus le nombre de lancers qu'on va devoir faire pour gagner. À quel genre de modification peut-on immédiatement penser lors de l'analyse de ce nouveau besoin ?

03.2 Quel genre de structure faudra-t-il utiliser pour traiter le cas spécial « on dépasse la case gagnante » ?

03.3 Expliquez comment vous allez traiter la partie « connaître le nombre de lancers qui ont été nécessaires pour gagner ».

03.4 Modifiez le code pour implémenter cette nouvelle version du jeu. Vous écrirez la totalité du nouveau programme sur votre copie, sauf les trois premières lignes, qui ne changeront pas.

```
Random generateur = new Random();
int caseGagnante = 20;
int maCase = 0;

for (int i = 1; i <= 5; i++) {
  int lancer = generateur.nextInt(6) + 1;
  System.out.println("Vous avez fait " + lancer);
  maCase = maCase + lancer;
  System.out.println("Vous êtes à la case " + maCase);
}
```

```
if (maCase == caseGagnante) {
  System.out.println("Gagné !");
} else {
  System.out.println("Perdu !");
}
```