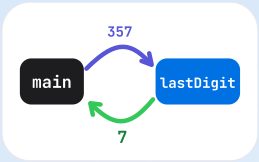
BagrutAce

علوم حاسوب · Java · 3 وحدات



الدوال methods
كيف تنادوا عليهن صح

تذكير سريع

بالصفحة 3 أسئلة من البنك، والحل الكامل بالصفحة 4 ←

- `static` بننادي عليها بدون كائن
- `int f(...)` بترجّع قيمة int
- `return` بترجّع الجواب وبتوقّف
- `(int n)` بالتعريف: parameter
- `void` بتعمل إشي، ما بترجّع
- `f(357)` بالنداء: argument

1 مبنى الدالة

4 أجزاء

`(int n)`	`lastDigit`	`int`	`public static`
الـ parameter: القيمة اللي بتستقبلها	اسم الدالة	نوع الجواب اللي بترجّعه	بننادي عليها بالاسم، بدون كائن

```java
public static int lastDigit(int n) {
  return n % 10;
}
```

كتبتوا الدالة؟ لسّا ما صار إشي. الكود جوّاها بيشتغل **بس ونتا حدا ينادي عليها**، و `return` بترجّع الجواب للمكان اللي نادى.

2 return ولا void ؟

بترجّع؟

```java
public static void printStars(int k) {
  for (int i = 0; i < k; i++)
    System.out.print("*");
  System.out.println();
}
```

`void` بتطبع وما بترجّع إشي، فبننادي عليها **كسطر لحاله**: `printStars(3);` ودالة `int` بترجّع رقم، والرقم لازم نستعمله.

3 نادوا واستعملوا الجواب

4 طرق

```java
int d = lastDigit(357);            // 7
System.out.println(lastDigit(92)); // 2
if (lastDigit(40) == 0)            // true
  System.out.println("ends with 0");
int sum = lastDigit(15) + lastDigit(28); // 13
printStars(3);                     // ***
```

النداء `lastDigit(357)` **بيصير قيمة** (7): بتنحط بمتغيّر، بشرط، بحساب أو بطباعة.

4 parameter و argument

```java
int x = 46;
int a = lastDigit(x + 1);   // n = 47, a = 7
```

`x + 1` هو الـ **argument**. قيمته 47 بتنتقل للـ **parameter** `n`، وبعدها بيشتغل الكود جوّا الدالة.

`lastDigit(x + 1)` → `n = 47` → `7`

5 دالة بتنادي دالة

```java
public static boolean sameLastDigit(int a,
                                    int b) {
  return lastDigit(a) == lastDigit(b);
}
```

`sameLastDigit(17, 207)` → `7 == 7` → `true`

«يجب الاستعانة بالعمليّة من البند أ» يعني: نادوا عليها، ما تكتبوها من جديد.

بالصفحة 2: من صياغة البچروت للكود + 3 أغلاط بالنداء ←

من صياغة البچروت للكود

6 شو كل كلمة بالسؤال بتعني؟

بتتكرّر بكل امتحان

«عمليّة خارجيّة باسم f»	`public static`
«تُعيد true / false»	`boolean + return`
«تطبع»	`void + println`
«تتلقّى عددًا صحيحًا num»	`(int num)`
«تُعيد العدد / المجموع»	`int + return`
«يجب الاستعانة بالعمليّة من البند أ»	`sumIsPrime(i)`

«تُعيد» و«تطبع» مش نفس الإشي. إذا السؤال بيحكي «تُعيد» وانتو طبعتوا، الجواب ما بيوصل للي نادى.

7 دالة بتنادي دالة مرّتين

```
public static int twice(int n) {
  return n * 2;
}
public static int addTwice(int a, int b) {
  return twice(a) + twice(b);
}
```

```
int r = addTwice(3, 4);      // 6 + 8 = 14
```

`addTwice(3, 4)` → `twice(3) + twice(4)` → `6 + 8` → `14`

8 من كلاس ثاني

static

```
public class Tools {
  public static int square(int n) {
    return n * n;
  }
}
```

```
int s = Tools.square(6);     // 36
```

بنفس الكلاس: الاسم بس. من كلاس ثاني: **Tools.square(...)**.
وstatic يعني ما بدنا كائن.

3 أغلاط بالنداء

1 الجواب راح

غلط ✗

```
lastDigit(357);
```

صح ✓

```
int kept = lastDigit(357);
```

بتشتغل وما في error، بس الـ 7 ما انحفظت بأي مكان. **حطّوها بمتغيّر** أو استعملوها.

2 void بمتغيّر

غلط ✗

```
int x = printStars(3);
```

صح ✓

```
printStars(3);
```

ما بتتقمبل (compile error): void ما بترجّع إشي. **نادوها كسطر لحاله.**

3 نوع جوّا النداء

غلط ✗

```
lastDigit(int 357);
```

صح ✓

```
int v = lastDigit(357);
```

النوع بينكتب **بس بالتعريف**. بالنداء بنبعث قيمة بس.

بالصفحة 3: 3 أسئلة من بنك BagrutAce . حلّوها لحالكم أوّل ←

3 أسئلة من البنك

BagrutAce

1 مجموعه أوّلي

متوسط

مجموع أرقام عدد موجب `num` هو مجموع أرقامه العشريّة. مثال: مجموع أرقام 29 هو 11، لأنّ 2+9=11. عدد موجب `num` يُسمّى **«مجموعه أوّليّ»** إذا كان مجموع أرقامه عددًا أوّليًّا. العدد الأوّليّ هو عدد أكبر من 1 ولا يقسم إلا على 1 وعلى نفسه (مثل 2 و3 و5 و7 و11). العدد 1 ليس أوّليًّا. أمثلة: 12 مجموعه أوّليّ (3=2+1)، و44 ليس مجموعه أوّليًّا (8=4+4)، و10 ليس مجموعه أوّليًّا (1=0+1).

أ. **اكتبوا عمليّة خارجيّة باسم `sumIsPrime`، تتلقّى عددًا صحيحًا `num` أكبر من 0. تُعيد العمليّة `true` إذا كان مجموع أرقام `num` عددًا أوّليًّا، وخلاف ذلك تُعيد `false`.**

ب. **اكتبوا عمليّة خارجيّة باسم `countSumPrime`، تتلقّى عددين صحيحين `low` و`high` أكبر من 0 (`low` أصغر من `high` أو مساوٍ له). تُعيد العمليّة كم عددًا مجموعه أوّليّ يوجد بين `low` و`high` (بما في ذلك الطرفان). يجب استعمال العمليّة من البند أ.**

2 عدد متعرّج

صعب

عدد موجب `num` أكبر أو يساوي 10 يُسمّى **«متعرّجًا»** إذا تحقّق الشرطان معًا: لا يوجد فيه رقمان متجاوران متساويان، والفروق بين كلّ رقمين متجاورين (من اليسار إلى اليمين) تتناوب إشارتها: صعود ثمّ هبوط ثمّ صعود، أو هبوط ثمّ صعود ثمّ هبوط. عدد من رقمين مختلفين يُعتبر متعرّجًا. أمثلة: 1324 متعرّج لأنّ 4 > 2 < 3 > 1. و121 متعرّج، و12 و10 متعرّجان. 123 ليس متعرّجًا (3 > 2 < 1 بنفس الاتّجاه)، و4321 ليس متعرّجًا، و11 و122 ليسا متعرّجين.

أ. **اكتبوا عمليّة خارجيّة باسم `isZigzag` تتلقّى عددًا صحيحًا `num` (أكبر أو يساوي 10)، وتُعيد `true` إذا كان متعرّجًا، وخلاف ذلك `false`.**

ب. **اكتبوا عمليّة خارجيّة باسم `printZigzag` تتلقّى عددًا صحيحًا `n` (أكبر أو يساوي 10)، وتطبع جميع الأعداد المتعرّجة من 10 حتى `n` (بما في ذلك 10 و`n`). مثلًا لـ n = 13 تطبع 10 و12 و13. يجب استعمال العمليّة من البند أ.**

جـ. **اكتبوا عمليّة خارجيّة باسم `largestZigzag` تتلقّى عددًا صحيحًا `n` (أكبر أو يساوي 10)، وتُعيد أكبر عدد متعرّج من 10 حتى `n`. مثلًا لـ n = 22 تُعيد 21 لأنّ 22 ليس متعرّجًا. يجب استعمال العمليّة من البند أ.**

3 متوازن الضبط

صعب

عدد موجب `num` يُسمّى **«متوازن الضبط»** إذا كان حاصل ضرب أرقامه يساوي مجموع أرقامه مضروبًا في عدد أرقامه، أي `product = sum * count`. مثلًا 36: المجموع 9 والحاصل 18 والعدد 2، و2 × 9 = 18 فيحقّق الشرط. 44 أيضًا يحقّق لأنّ 2 × 8 = 16. 25 لا يحقّق لأنّ 10 لا يساوي 14 = 2 × 7.

أ. **اكتبوا 4 عمليّات: (1) `digitSum` تُعيد مجموع أرقام `num`. (2) `digitProduct` تُعيد حاصل ضرب أرقامه. (3) `digitCount` تُعيد عدد أرقامه. (4) `isProdSumCount` تُعيد `true` إذا كان `product = sum * count`، وخلاف ذلك `false`. يجب الاستعانة بالعمليّات الثلاث من (1) حتى (3).**

ب. **اكتبوا عمليّة خارجيّة باسم `theLargest` تتلقّى عددًا صحيحًا موجبًا `n`، وتطبع أكبر عدد حتى `n` (بما في ذلك `n`) يحقّق الشرط. كلّ الأعداد من 1 حتى 9 تحقّق الشرط، لذلك يوجد حلّ دائمًا. يجب الاستعانة بـ `isProdSumCount` بالبحث نزولًا من `n`. مثلًا لـ n = 50 تطبع 44، ولـ n = 10 تطبع 9.**

✓ طريقة الحل بـ 3 خطوات

1. اكتبوا الدالة الصغيرة (البند أ) وجرّبوها على المثال اللي بالسؤال.
2. بالبنود اللي بعدها نادوا عليها، ما تنسخوا الكود.
3. «تُعيد» = return، «تطبع» = println. اقرأوا الفعل منيح.

الأسئلة الثلاثة من بنك أسئلة BagrutAce، بنفس صياغة البچروت. حلّوها على ورقة، والحل الكامل بالصفحة 4.

1 مجموعه أوّلي

أ دالتين مساعدات، و sumIsPrime بتنادي عليهن

```java
public static int digitSum(int num) {
  int sum = 0;
  while (num > 0) {
    sum += num % 10;
    num /= 10;
  }
  return sum;
}
public static boolean isPrime(int n) {
  if (n < 2)
    return false;
  for (int d = 2; d < n; d++)
    if (n % d == 0)
      return false;
  return true;
}
```

```java
public static boolean sumIsPrime(int num) {
  return isPrime(digitSum(num));
}
```

ب countSumPrime

```java
public static int countSumPrime(int low,
                                int high) {
  int count = 0;
  for (int i = low; i <= high; i++)
    if (sumIsPrime(i))
      count++;
  return count;
}
```

`digitSum(29)` → `11` → `isPrime(11)` → `true`

i	10	11	12	13	14	15
digitSum(i)	1	2	3	4	5	6
sumIsPrime(i)	✗	✓	✓	✗	✓	✗

3 countSumPrime(10, 15): الأعداد 11 و12 و14. وisPrime بترجّع false لـ 1، فـ 10 ما بينعدّ.

2 عدد متعرّج

أ isZigzag : بنقرأ الأرقام من اليمين

```java
public static boolean isZigzag(int num) {
  int prev = num % 10;   // last digit
  num /= 10;
  int lastDiff = 0;
  while (num > 0) {
    int d = num % 10;
    int diff = d - prev;
    if (diff == 0 || diff * lastDiff > 0)
      return false;   // equal / same sign
    lastDiff = diff;
    prev = d;
    num /= 10;
  }
  return true;
}
```

ب printZigzag : void ، بتطبع

```java
public static void printZigzag(int n) {
  for (int i = 10; i <= n; i++)
    if (isZigzag(i))
      System.out.println(i);
}
```

ج largestZigzag : int ، بترجّع

```java
public static int largestZigzag(int n) {
  for (int i = n; i >= 10; i--)
    if (isZigzag(i))
      return i;
  return 10;
}
```

true 1324 من اليمين: 4، 2، 3، 1. الفروق 2−، 1+، 2−: الإشارة بتتبدّل كل مرّة.

false 123: الفروق 1−، 1− بنفس الإشارة، فحاصل ضربهم موجب.

21 largestZigzag(22): بنبلّش من 22 ونازلين. أوّل عدد متعرّج بنلاقيه هو الجواب، وreturn بتوقّف اللفّة.

diff * lastDiff > 0 يعني نفس الإشارة مرّتين ورا بعض. والسؤال كله بيرجع لـ isZigzag: ب وجـ بس بينادوا عليها.

3 متوازن الضبط

أ (3-1) 3 دوال صغيرة، نفس اللفّة على الأرقام

```
public static int digitSum(int num) {
  int sum = 0;
  while (num > 0) {
    sum += num % 10;
    num /= 10;
  }
  return sum;
}
public static int digitProduct(int num) {
  int prod = 1;
  while (num > 0) {
    prod *= num % 10;
    num /= 10;
  }
  return prod;
}
public static int digitCount(int num) {
  int count = 0;
  while (num > 0) {
    count++;
    num /= 10;
  }
  return count;
}
```

أ (4) بتنادي على الثلاث وبتحفظ الأجوبة

```
public static boolean isProdSumCount(int num) {
  int s = digitSum(num);
  int p = digitProduct(num);
  int c = digitCount(num);
  return p == s * c;
}
```

ب theLargest : بتطبع، فـ void

```
public static void theLargest(int n) {
  int cand = n;
  while (!isProdSumCount(cand))
    cand--;
  System.out.println(cand);
}
```

prod بيبلّش من 1 مش من 0، وإلّا الحاصل دايمًا 0. واللفّة بتوقف أكيد، لأنّ 1 حتى 9 بيحقّقوا الشرط.

ب شو بيصير بـ theLargest(50) ؟

cand	digitSum	digitProduct	digitCount	s * c	p == s * c?
50	5	0	2	10	✗
49	13	36	2	26	✗
48	12	32	2	24	✗
47	11	28	2	22	✗
46	10	24	2	20	✗
45	9	20	2	18	✗
44	8	16	2	16	✓

44 بتطبع 44. كل لفّة فيها نداء لـ isProdSumCount، وهي بتنادي على 3 دوال.

9 theLargest(10): 10 ما بيحقّق (الحاصل 0)، فبتنزل لـ 9.

`theLargest(50)` → `isProdSumCount(44)` → `digitSum(44) = 8` `digitProduct(44) = 16` `digitCount(44) = 2` → `16 == 8 * 2`

بدكم تتدرّبوا على دوال؟

حلّوا على BagrutAce وAce بيصلّحلكم على طول.

bagrutace.com ← يلّا نبلّش