جافا - الوحدة الخامسة: المصفوفات والقوائم والخرائط

تتعامل البرامج الحقيقية مع قيم كثيرة: كل المنتجات، وكل العملاء، وكل طلبات اليوم. والمتغيرات المنفصلة لا تتسع لذلك، فتقدم جافا بنى تحمل قيما كثيرة معا. وتغطي أربع خطوات الخيارات اليومية: المصفوفة، والقائمة، والخريطة، واختيار الأنسب. وفي هذه الوحدة يستخدمها عمر، مدير مخزون في سلسلة متاجر للأجهزة الإلكترونية، ليتتبع مخزون كل منتج، وتجيب كل خطوة عن سؤال عملي واحد.

1. المصفوفة

تحمل المصفوفة عددا ثابتا من القيم من النوع نفسه. نحدد حجمها عند إنشائها، ولا يتغير الحجم بعد ذلك. ونصل إلى كل قيمة بفهرسها، وأول فهرس هو الصفر وليس الواحد. وطول المصفوفة متاح بالخاصية length دون أقواس. والطلب بفهرس خارج المصفوفة يوقف البرنامج بالخطأ ArrayIndexOutOfBoundsException، وهو من أكثر أخطاء المبتدئين شيوعا. وسؤال هذه الخطوة هو: هل نعرف بالضبط كم قيمة ستكون؟

مثال: تحفظ عيادة ساعات دوامها لأيام الأسبوع السبعة في مصفوفة من سبع قيم. والخانة الأولى فهرسها صفر، فيقع يوم الاثنين عند الصفر ويوم الأحد عند الستة. وليس للأسبوع يوم ثامن، لذلك يناسبه حجم ثابت تماما. والطلب بالفهرس سبعة يوقف البرنامج، لأن المصفوفة تنتهي عند الفهرس ستة.

في العمل: يحفظ عمر كمية منتج واحد في كل من مستودعات المناطق الخمسة في مصفوفة من خمسة أعداد. وتزور الحلقة الفهرس صفرا إلى الفهرس أربعة، والحلقة التي تتقدم خطوة زائدة تتعطل بخطأ الفهرس. والمصفوفة مناسبة هنا لأن المستودعات الخمسة لا تتغير. وفي الشهر التالي يفتتح مستودع سادس، فيرى حد الحجم الثابت.

2. القائمة

القائمة تسلسل يتغير حجمه ويحفظ الترتيب الذي أضيفت به القيم. وفي جافا اليومية يشيع التنفيذ ArrayList الموجود في الحزمة java util، ونحضره إلى ملفنا بسطر import. ونعلن نوع عناصرها بين أقواس زاوية، وتسمى generics، مثل قائمة من النوع String، فيرفض المترجم النوع الخاطئ. ومن الطرق المفيدة add و get و size و remove و contains، وتمر حلقة for each على كل العناصر. وتحمل القائمة كائنات، فقائمة الأعداد الصحيحة تستخدم الغلاف Integer، وتحوله جافا تلقائيا. ونعلن المتغير عادة بالواجهة List وننشئ ArrayList، فتبقى شيفرتنا مرنة. وسؤال هذه الخطوة هو: هل يتغير عدد القيم، وهل يهم الترتيب؟

مثال: يحتفظ مكتب دعم بقائمة بالتذاكر المفتوحة بحسب وصولها. وتضاف التذاكر الجديدة في النهاية بالطريقة add، وتحذف التذاكر التي حلت بالطريقة remove. وتكبر القائمة في صباح مزدحم تصل فيه تذاكر كثيرة، وتصغر من جديد عندما ينجز الفريق التذاكر المتراكمة. وتبقى الشيفرة هي نفسها في الحالتين، لأن القائمة لا تحتاج إلى حجم ثابت.

في العمل: يستبدل عمر بالمصفوفة قائمة كميات، لأن المستودعات تفتتح وتغلق. ويضيف الكمية السادسة باستدعاء واحد للطريقة add، وتجمع حلقة for each المخزون. وعندما يغلق مستودع، يخرجه استدعاء remove من القائمة. ويرفض المترجم محاولة إضافة نص إلى قائمة أعداد، وهذا يحميه من خطأ صامت.

3. الخريطة

تخزن الخريطة أزواجا: مفتاحا وقيمته. والوصول إلى القيمة بمفتاحها مباشر، فيبقى سريعا حتى عندما تكون الخريطة كبيرة. والتنفيذ الشائع هو HashMap. وطرقها الرئيسية put لتخزين زوج، و get لقراءة قيمة، و containsKey

حقوق الطبع والنشر محمية © - أكاديمية أساسيات (شركة أساسيات للخدمات الاستشارية)

للسؤال عن وجود مفتاح، و remove. والمفاتيح فريدة، فتخزين زوج بمفتاح موجود يستبدل القيمة القديمة، وتعيد get القيمة null عندما يغيب المفتاح. ولا تعد HashMap بأي ترتيب، فإذا كان الترتيب مهما اخترنا LinkedHashMap أو TreeMap. وسؤال هذه الخطوة هو: هل نجد كل قيمة باسم أو رمز وليس بالموضع؟

مثال: تخزن جامعة اسم كل طالب تحت رمز الطالب. ويعيد استدعاء الطريقة get بالرمز الاسم فورا، دون بحث بين بقية الطلاب. وإذا لم يكن الرمز في الخريطة أعاد الاستدعاء null، لذلك يجب أن يفحص البرنامج قبل الاستخدام. واستدعاء طريقة على القيمة null يوقف البرنامج بخطأ.

في العمل: تعرف منتجات عمر برمز منتج، فيحتاج إلى معرفة مخزون أي رمز في لحظة. ويعلن الخريطة بالسطر `Map<String, Integer> stock = new HashMap<>();`، والرمز هو المفتاح والكمية هي القيمة. ويصير استلام البضائع قراءة للكمية القديمة يتبعها put بالإجمالي الجديد. وقبل قراءة رمز غير معروف يسأل containsKey، فلا يفاجئه منتج غائب.

4. الاختيار

تتبع البنية الصحيحة السؤال الذي نطرحه. فإذا كان عدد القيم ثابتا ومعروفا فالمصفوفة بسيطة ومدمجة. وإذا كان العدد يتغير والترتيب مهما فالقائمة هي الخيار الطبيعي. وإذا بحثنا عن القيم بمفتاح فالخريطة هي الأداة، وإذا أردنا فقط أن نعرف هل الشيء موجود دون تكرار فالمجموعة مثل HashSet تقوم بالمهمة. والجمع بينها أمر طبيعي: فخريطة من رمز المنتج إلى قائمة حركات شكل شائع. وسؤال هذه الخطوة هو: كيف سنطلب البيانات في أغلب الأحيان؟

مثال: تحتاج وكالة سفر إلى التحقق سريعا مما إذا كان رقم جواز قد عولج من قبل. وتجيب المجموعة set عن السؤال مباشرة باستدعاء واحد للطريقة contains، ولا تحتفظ بالتكرار. أما القائمة فتحتاج إلى بحث في كل عنصر، وأما الخريطة فتطلب قيمة لا يريد أحد تخزينها. والمجموعة هي الأداة الصحيحة لأن السؤال الوحيد هو الوجود.

في العمل: يعتمد تصميم عمر النهائي خريطة من رمز المنتج إلى كمية المخزون. وتربط خريطة ثانية كل رمز بقائمة من حركات مخزونه بالترتيب الذي حدثت به. وتحمل مجموعة set رموز المنتجات المتوقفة، فيفحصها في خطوة واحدة. وكل اختيار يتبع طريقة طرحه لأسئلته.

الخاتمة

في الخلاصة، تحمل المصفوفات عددا ثابتا من القيم يوصل إليها بفهرس يبدأ من الصفر. وتكبر القوائم وتصغر وتحفظ الترتيب، وتجد الخرائط القيمة بمفتاحها. ونختار بالسؤال الذي نطرحه أكثر من غيره، وتجيب المجموعة set عن هل هذا موجود دون تكرار. وبرنامج المخزون عند عمر يتعامل الآن مع أي عدد من المستودعات ويجد المنتج برمزه. وفي المرة القادمة سنصف المنتجات والموظفين بالأصناف والكائنات. ومن العادات الجيدة إعلان المتغير بنوع الواجهة مثل List أو Map، وتحديد التنفيذ في مكان واحد فقط.

المصادر

- Oracle: دروس جافا، المصفوفات – https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html
- Oracle: دروس جافا، مسار المجموعات (القوائم والمجموعات والخرائط) – https://docs.oracle.com/javase/tutorial/collections/index.html
- Oracle: دروس جافا، الأنواع العامة Generics – https://docs.oracle.com/javase/tutorial/java/generics/index.html
- dev.java: تعلّم جافا – https://dev.java/learn/

• Java علامة تجارية لشركة Oracle و/أو الشركات التابعة لها. هذا الفيديو مادة تدريبية مستقلة وغير تابعة لـ Oracle.

حقوق الطبع والنشر محمية © - أكاديمية أساسيات (شركة أساسيات للخدمات الاستشارية)