جافا - الوحدة السادسة: الأصناف والكائنات

استخدمت برامجنا حتى الآن قيما منفصلة، لكن الأعمال تفكر بالأشياء: موظف، فاتورة، عميل. ويصف الصنف شيئا كهذا مرة واحدة، ويحمل كل كائن يصنع منه بياناته الخاصة. وتبني أربع خطوات صنفا جيدا: المخطط، والبناء، والحماية، والسلوك. وفي هذه الوحدة تستخدمها سلمى، أخصائية موارد بشرية في شركة تصنيع، لتمثل الموظفين في الشيفرة، وتجيب كل خطوة عن سؤال عملي واحد.

1. المخطط

الصنف مخطط يسجل ما يعرفه الشيء وما يستطيع فعله. وما يعرفه يخزن في حقول، وهي متغيرات داخل الصنف. وما يستطيع فعله يكتب طرقا. والكائن شيء واحد يصنع من الصنف بالكلمة new، ولكل كائن نسخته الخاصة من قيم الحقول. والمتغير الذي يحمل كائنا يحمل في الحقيقة مرجعا إليه، فقد يشير متغيران إلى الكائن نفسه، والمتغير الذي لا يشير إلى شيء قيمته null. واستدعاء طريقة على null يوقف البرنامج بالخطأ NullPointerException. وسؤال هذه الخطوة هو: ماذا يعرف هذا الشيء، وماذا يستطيع أن يفعل؟

مثال: تصف شركة تأجير سيارات المركبة مرة واحدة، في صنف له ثلاثة حقول: لوحة وطراز وسعر يومي. ومن هذا المخطط الواحد تنشئ كل سيارة في أسطولها بالكلمة new. ويحمل كائن السيارة الأولى لوحة وسعرا، ويحمل كائن السيارة الثانية لوحة وسعرا مختلفين. وتغيير السعر اليومي لسيارة واحدة لا يمس باقي السيارات، لأن كل كائن يحتفظ بنسخته الخاصة من الحقول.

في العمل: تحتفظ سلمى ببيانات الموظفين في ثلاث قوائم متوازية: الأسماء والرواتب والأقسام. وعندما يغادر موظف يحذف اسم من قائمة ولا يحذف من القائمتين الأخريين، فتنزلق الرواتب إلى أسماء خاطئة. فتنشئ صنف Employee بحقول للاسم والقسم والراتب. ويصير كل موظف الآن كائنا واحدا، ولا تتباعد بيانات شخص واحد أبدا.

2. البناء

المنشئ طريقة خاصة تعمل عند إنشاء كائن بالكلمة new. وله اسم الصنف نفسه ولا نوع قيمة معادة له، ومهمته إعطاء الحقول قيمها الأولى. وتشير الكلمة this إلى الكائن الذي يبنى أو يستخدم، فنميز بها الحقل من المعامل الذي له الاسم نفسه. وإذا لم نكتب منشئا زودتنا جافا بمنشئ فارغ، لكن بمجرد كتابة منشئنا يختفي الفارغ ما لم نعده. ويستطيع الصنف أن يمتلك عدة منشئات بمعاملات مختلفة، تماما مثل الطرق المحملة. والمنشئ الذي يطلب البيانات الأساسية يجعل إنشاء كائن ناقص مستحيلا. وسؤال هذه الخطوة هو: ماذا يجب أن يملك الكائن قبل أن يوجد؟

مثال: تشترط مكتبة عنوانا ومؤلفا قبل أن تنشئ سجل كتاب. ويأخذ منشئ الصنف Book القيمتين معاملين ويخزنهما في الحقلين بالكلمة this. وسطر يحاول إنشاء كتاب بلا عنوان لا يترجم، فلا يمكن أن يوجد سجل ناقص. وتبقى الشيفرة التي تنشئ الكتب قصيرة وواضحة، لأن سطرا واحدا يعطي الكتاب كل ما يحتاج إليه.

في العمل: تقرر سلمى أن للموظف دائما اسما وقسما وراتبا. وتكتب منشئا بثلاثة معاملات وتستخدم this لتخزين كل منها في حقله. ومحاولة إنشاء Employee بلا اسم لا تترجم بعد الآن. ويأخذ منشئ ثان الاسم فقط ويضع قسما افتراضيا للموظفين الجدد.

3. الحماية

التغليف encapsulation يعني إخفاء الحقول والتحكم في الوصول إليها عبر الطرق. نجعل الحقول private فلا يلمسها إلا الكود داخل الصنف، ونقدم طرقا public لقراءتها أو تغييرها. والطريقة التي تقرأ حقلا تسمى getter، والتي تغيره تسمى setter. ولأن كل تغيير يمر بطريقة، يستطيع الصنف فحص القيمة أولا ورفض راتب سالب. وهذا يبقي قواعد

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

الكائن في مكان واحد، بدلا من الثقة بأن يكون كل مستدع حذرا. ولا يحتاج كل حقل إلى setter: فالحقل الذي لا يتغير بعد الإنشاء، مثل رقم الموظف، يكفيه getter فقط. وسؤال هذه الخطوة هو: من يحق له قراءة هذا الحقل أو تغييره، وبأي قواعد؟

مثال: يبقي صنف حساب مصرفي رصيده private. والطريقتان الوحيدتان لتغيير الرصيد هما الإيداع والسحب، وترفض طريقة السحب مبلغا أكبر من الرصيد. وإذا حاول جزء آخر من البرنامج ضبط الرصيد مباشرة فلن تترجم الشيفرة. فلا يستطيع أي كود آخر في البرنامج كسر هذه القاعدة، وتكتب القاعدة في مكان واحد فقط.

في العمل: تجعل سلمى الحقول private، مثل `private BigDecimal salary;`، وتضيف getters لها كلها. ولا تكتب setter عاديا للراتب، بل تضيف طريقة اسمها changeSalary ترفض المبلغ السالب. ويأخذ رقم الموظف getter فقط، لأنه لا يتغير أبدا. ويستطيع زملاؤها الآن استخدام Employee دون أن يكسروا قواعده.

4. السلوك

تعمل الطرق داخل الصنف على بيانات كائن واحد. وتستخدم طرق الكائن هذه الحقول مباشرة، فطريقة مثل raiseSalary لا تحتاج إلا إلى النسبة، وتغير راتب الكائن الذي استدعيت عليه. وتعيد الطريقة toString وصفا مقروءا، وتستدعيها جافا تلقائيا عند طباعة كائن أو وصله بنص. والطريقة الثابتة static أو الحقل الثابت ينتميان إلى الصنف نفسه لا إلى كائن واحد، مثل عداد لعدد الموظفين الذين أنشئوا. ونحفظ الكائنات معا في مجموعات، مثل قائمة من Employee، فتطلب حلقة for each من كل كائن أن يؤدي عمله. والأصناف الجيدة صغيرة، لها اسم واضح هو اسم لشيء، وطرق تقرأ كأفعال. وسؤال هذه الخطوة هو: ماذا ينبغي أن يستطيع هذا الكائن فعله بنفسه؟

مثال: يصف متجر منتجا باسم وسعر وطريقة تطبق خصما. وعندما تستدعى الطريقة على منتج واحد فإنها تغير سعر ذلك المنتج فقط، وتحتفظ المنتجات الأخرى بأسعارها. وتظهر طباعة المنتج اسمه وسعره بنص مقروء بفضل toString. وكود المتجر يطلب من كل منتج أن يطبق الخصم على نفسه، ولا يحسب السعر من الخارج أبدا.

في العمل: تضيف سلمى طريقة raiseSalary تأخذ نسبة وتحدث راتب ذلك الموظف. وتضيف toString لتظهر طباعة الموظف اسمه وقسمه وراتبه. ويحتفظ فريق الرواتب بقائمة من كائنات Employee، وتطلب حلقة for each من كل واحد تطبيق الزيادة السنوية. ويحتاج تقرير القسم كله الآن إلى أسطر قليلة بدلا من ثلاث قوائم متوازية.

الخاتمة

في الخلاصة، الصنف مخطط بحقول وطرق، والكائن شيء واحد يصنع منه وله بياناته الخاصة. والمنشئ يضمن أن يبدأ كل كائن كاملا، والكلمة this تفصل الحقل عن المعامل. والحقول الخاصة مع طرق منظمة تحمي قواعد الكائن، ويقع السلوك داخل الكائن نفسه. وموظفو سلمى ينتقلون الآن كائنات مفردة تحفظ بياناتها معا. وفي المرة القادمة سنرى كيف تتشارك الأصناف السلوك بالوراثة والواجهات. ومن العادات الجيدة تصميم الصنف من أسماء مشكلة العمل وطرقه من أفعالها.

المصادر

- Oracle: دروس جافا، الأصناف والكائنات – https://docs.oracle.com/javase/tutorial/java/javaOO/index.html
- Oracle: دروس جافا، توفير المنشئات لأصنافك – https://docs.oracle.com/javase/tutorial/java/javaOO/constructors.html
- Oracle: دروس جافا، التحكم في الوصول إلى أعضاء الصنف – https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.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)