جافا - الوحدة السابعة: الوراثة والواجهات وتعدد الأشكال

تأتي مفاهيم الأعمال في عائلات: أنواع كثيرة من المطالبات أو الحسابات أو المنتجات تشترك في جوهر وتختلف في التفاصيل. وتسمح جافا للأصناف بمشاركة السلوك دون نسخ الكود. وتغطي ذلك أربع خطوات: الوراثة، وإعادة التعريف، والعقد، والإحلال. وفي هذه الوحدة يستخدمها يوسف، محلل مطالبات في شركة تأمين، ليعالج مطالبات السيارات والعقارات والصحة بكود واحد، وتجيب كل خطوة عن سؤال عملي واحد.

1. الوراثة

تسمح الوراثة لصنف جديد، هو الصنف الفرعي، أن يبدأ من صنف قائم، هو الصنف الأصلي، بالكلمة extends. ويتسلم الفرعي الحقول والطرق التي يسمح له الأصلي برؤيتها، ويضيف إليها ما يخصه. وتبقى الأعضاء private مختبئة داخل الأصلي، وهذا سبب آخر لإعطائها getters. ولا يستطيع الصنف أن يرث إلا من صنف واحد، وكل صنف يرث في النهاية من Object، ومنه تأتي toString. ويبدأ منشئ الصنف الفرعي عادة باستدعاء منشئ الأصلي بالكلمة super، فتبنى البيانات المشتركة أولا. وتناسب الوراثة علاقة حقيقية من نوع هو نوع من: مطالبة السيارة نوع من المطالبات. وسؤال هذه الخطوة هو: ماذا تشترك فيه هذه الأنواع حقا؟

مثال: لدى مصرف حسابات ادخار وحسابات جارية. ولكليهما مالك ورصيد، فيضع المطور هذين الحقلين في صنف Account واحد. ثم يرث صنف حساب الادخار من Account ويضيف معدل الفائدة، ويضيف صنف الحساب الجاري حد السحب على المكشوف. فيعيد كل نوع استخدام المالك والرصيد ويضيف ما يميزه فقط.

في العمل: يجد يوسف ثلاثة أصناف لمطالبات السيارات والعقارات والصحة، في كل منها رقم المطالبة والمطالب والتاريخ نفسها. فينقل هذه الحقول المشتركة و getters الخاصة بها إلى صنف Claim. وترث MotorClaim و PropertyClaim و MedicalClaim الآن من Claim وتضيف تفاصيلها فقط، كمركبة أو رمز تشخيص. ويبدأ كل منشئ باستدعاء super يمرر البيانات المشتركة إلى الأعلى.

2. إعادة التعريف

يستطيع الصنف الفرعي أن يستبدل نسخته الخاصة بطريقة ورثها، ويسمى هذا override. وللطريقة الجديدة الاسم والمعاملات نفسها، ونضع عليها التعليق التوضيحي Override ليتأكد المترجم من أننا نستبدل شيئا موجودا فعلا. وداخل الطريقة الجديدة تستدعي الكلمة super متبوعة باسم الطريقة نسخة الأصلي، وهذا مفيد عندما نريد الإضافة فقط إلى سلوك الأصلي. وعند استدعاء الطريقة تشغل جافا النسخة التي تخص الكائن الفعلي، حتى لو كان نوع المتغير هو الأصلي. والطريقة final لا يمكن إعادة تعريفها، والصنف final لا يمكن أن يرث منه صنف. ويختلف هذا عن التحميل الزائد، لأن المعاملات تبقى كما هي وتقرر عائلة الأصناف أي نسخة تعمل. وسؤال هذه الخطوة هو: ماذا يفعل هذا النوع بصورة مختلفة؟

مثال: لدى شركة شحن طريقة تحسب رسوم أي طرد من وزنه. وللطرود القابلة للكسر يعيد الصنف الفرعي تعريف هذه الطريقة، فيستدعي نسخة الأصلي أولا ثم يضيف رسم مناولة فوقها. ويستخدم الطرد العادي نسخة الأصلي. وينبه التعليق Override أيضا إلى خطأ إملائي في اسم الطريقة.

في العمل: لكل مطالبة طريقة payout، لكن كل نوع يحسبها بصورة مختلفة. فتخصم MotorClaim المبلغ الذي يتحمله العميل من تكلفة الإصلاح، وتطبق MedicalClaim نسبة التغطية على الفاتورة. ويضع يوسف Override على كل نسخة، فيلتقط المترجم فورا أي خطأ في اسم الطريقة. وتستدعي PropertyClaim الدفع الأساسي للأصلي أولا ثم تضيف بدل المحتويات.

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

3. العقد

الواجهة عقد يسجل طرقا يعد الصنف بتوفيرها دون أن يقول كيف. ويوقع الصنف العقد بالكلمة implements، ويتأكد المترجم من أن كل طريقة مكتوبة موجودة. ويستطيع الصنف أن يرث من صنف واحد فقط لكنه ينفذ واجهات كثيرة، وهذا يجعل الواجهات الأداة المرنة. ويقف الصنف المجرد abstract بين الاثنين: لا يمكن إنشاء كائن منه، وقد يعلن طرقا مجردة يجب أن ينفذها كل صنف فرعي، بينما يشارك الكود في الباقي. وتصف الواجهات ما يستطيع الشيء فعله، أما الوراثة فتصف ما هو. والواجهات الصغيرة ذات الغرض الواضح أسهل في التنفيذ وفي الاستبدال. وسؤال هذه الخطوة هو: ماذا يجب أن يعد به كل نوع، مهما كانت الطريقة؟

مثال: يعرف نظام دفع عقدا اسمه PaymentMethod فيه طريقة واحدة تخصم مبلغا. وتنفذ البطاقات والتحويلات المصرفية والمحافظ العقد كل بطريقته، فتتصل البطاقة بالمصرف بينما تخفض المحفظة رصيدا مخزنا. ولا يعرف كود الدفع النهائي إلا العقد، فيطلب من أي وسيلة خصم المبلغ. وتنضم وسيلة جديدة بتوقيع العقد، ولا يتغير كود الدفع النهائي.

في العمل: يستخرج يوسف قاعدة الدفع في عقد: `interface Payable { BigDecimal payout(); }`. وينفذ الصنف Claim الواجهة Payable، فيجب أن يوفر كل نوع مطالبة طريقة payout. ولاحقا ينفذها صنف المبالغ المستردة لدى الفريق المالي أيضا، مع أنه ليس مطالبة. ويرفض المترجم أي صنف يوقع العقد وينسى الطريقة.

4. الإحلال

تعدد الأشكال polymorphism يعني أن متغيرا من نوع أصلي أو من نوع واجهة يستطيع حمل أي كائن يناسبه. ونكتب الكود للنوع العام، وعند التشغيل تختار جافا نسخة الطريقة التي تخص الكائن الفعلي. فتستطيع حلقة واحدة على قائمة مطالبات جمع مدفوعات مطالبات السيارات والعقارات والصحة دون أن تعرف نوع كل مطالبة. وعندما يظهر نوع جديد نضيف صنفا ويبقى الكود القائم يعمل. ونحتاج أحيانا إلى السؤال عما هو الكائن فعلا بالكلمة instanceof، لكن كثرة الحاجة إليها علامة تحذير. وسؤال هذه الخطوة هو: هل يستطيع الكود المستدعي أن يعمل بالنوع العام وحده؟

مثال: تشغل منصة فيديو ملفات بصيغ كثيرة عبر واجهة واحدة اسمها Player. ويستدعي كود الشاشة play على Player ولا يفحص صيغة الملف أبدا. وينفذ صنف كل صيغة الطريقة play بطريقته، وتشغل جافا النسخة الصحيحة. وإضافة صيغة جديدة تعني إضافة صنف واحد، ولا يتغير كود الشاشة.

في العمل: يحفظ يوسف كل المطالبات المفتوحة في قائمة من Payable ويمر عليها بحلقة for each. وتستدعي الحلقة payout على كل عنصر، وتختار جافا الحساب الصحيح لكل كائن. وعندما تطلق الشركة مطالبات التأمين السيبراني يضيف صنفا واحدا ولا يغير سطرا من الحلقة. ويجمع تقرير نهاية الشهر كل المدفوعات في مرور واحد.

الخاتمة

في الخلاصة، تسمح الوراثة لصنف فرعي بإعادة استخدام صنف أصلي، ويسمح override له بفعل شيء واحد بصورة مختلفة. وتصف الواجهات ما يعد الصنف بفعله، ويجمع الصنف المجرد بين الكود المشترك والطرق المطلوبة. ويسمح تعدد الأشكال للكود بالعمل مع النوع العام بينما تختار جافا النسخة الصحيحة عند التشغيل. وتقرير نهاية الشهر عند يوسف يعالج الآن كل أنواع المطالبات بحلقة واحدة، وتنضم الأنواع الجديدة دون لمسها. وفي المرة القادمة سنتعامل مع الأخطاء: الاستثناءات وقراءة الملفات. ومن العادات الجيدة تفضيل الواجهات الصغيرة واستخدام الوراثة فقط لعلاقة حقيقية من نوع هو نوع من.

المصادر

- Oracle: دروس جافا، الوراثة – https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html
- Oracle: دروس جافا، الواجهات – https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html
- Oracle: دروس جافا، تعدد الأشكال – https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.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)