التطبيقات الموزعة – جلسة 2 – مراجعة جافا 1

أهداف الجلسة

هذه الجلسة مخصصة لمراجعة بعض المفاهيم الأساسية في لغة البرمجة جافا بهدف تقوية الطلاب في هذه اللغة و استخدامها بشكل فعال ضمن الجلسات القادمة كما تتضمن مثالين حول نمذجة بعض الحالات باستخدام البرمجة كائنية التوجه.

من المتوقع في نهاية الجلسة أن يكون الطالب قادر على:

- تمييز أنواع البيانات الأساسية في لغة البرمجة جافا و تعريف المتحولات و استخدام العوامل عليها.
- كتابة العبارات الشرطية و الحلقات التكرارية.
- تعريف الصفوف و الطرائق و الواصفات ضمن الصفوف.
- استخدام الوراثة لبناء العلاقات بين الصفوف.
- حل أمثلة باستخدام الصفوف و رفع الحلول إلى مشروع خاص private على github.

الأنواع في لغة البرمجة جافا

تقسم الأنواع في جافا إلى نمطين رئيسيين هما

1. أنواع البيانات الأولية primitive data types: هذه الأنواع هي معرفة بشكل مسبق في لغة البرمجة جافا, هي ثمانية أنواع: byte, short, int, long, float, double, boolean and char.
2. أنواع البيانات المرجعية Reference data types: هذه الأنواع يتم تعريفها باستخدام الصفوف و تدعى المتحولات التي تأخذها كائنات من صف ما يتم فيها إستخدام مؤشر إلى القيمة الحقيقة.

الثوابت في جافا

يتم اسخدام الثوابت literals لإستاد قيمة إلى المتحولات من أنواع البيانات الأولية فقط, حيث يتم إسناد هذه القيم أثناء عملية ترجمة الملف المصدري أمثلة عن هذه الثوابت

12, 561L, ‘j’, 0xfa7, 012, 12.3f, 90.123d, 0.9e-12, etc….

المتحولات

يتم تعريف المتحولات في جافا باستخدام الصيغة التالية: type varname [=value];

يوجد ثلاثة أنواع للمتحولات في جافا:

1) المتحولات المحلية: يتم تعريفها ضمن الطرائق, الباني و الكتل blocks لا تمتلك محددات وصول access modifiers كمان أنها لا تمتلك أي قيمة إفتراضية, تبقى معرفة فقط ضمن الكتلة التي تم تعريفها فيها.
2) متحولات الكائنات instance variables: يتم تعريفها ضمن الصفوف خارج الطرائق, تمتلك محددات وصول و تكون قيمتها مرتبطة بالكائن الذي تم تعريفه, تكون معرفة من لحظة بناء الكائن إلى لحظة تدميره.
3) متحولات الصفوف class or static variables: تعرف هذه المتحولات ضمن الصفوف باستخدام الكلمة المفتاحية static, تمتلك قيمة افتراضية كما أنها تبقى معرفة من لحظة تشغيل البرنامج إلى لحظة إنتهائه و تكون قيمتها مشتركة بين جميع الكائنات في الصف الذي عرفت ضمنه.

العوامل operators

يوجد ستة أصناف للعوامل ضمن جافا هي:

1) العوامل الحسابية +, -, *, /, %, ++, --.
2) عوامل المقارنة ==, !=, >, >=, <, <=.

3) العوامل المختصة بالبت <<< ,>> ,<< ,~ ,^ ,| ,&.
4) العوامل المنطقية: ! ,|| ,&&.
5) عوامل الإسناد: =^ ,=| ,=& ,>>= ,<<= ,%= ,\= ,*= ,-= ,+= ,+.
6) عوامل أخرى: instanceof ,:?.

العبارات الشرطية

يتم استخدام عبارات if الشرطية لتنفيذ جزء من الكود بناءً على شرط معين:

```
if (condition) {

}

else if (condition){

}

else {

}
```

يمكن إستخدام أكثر من عبارة if لتحديد أكثر من شرط كمان يمكن استخدام عبارة else لتنفيذ الكود في حال لم يكن أي شرط محقق.

يمكن استخدام المعامل :? بدلاً من عبارتين if/else لإسناد قيمة لمتحول ما بناء على صحة شرط أو عدم صحته.

العبارة switch يمكن أن تستخدم بدلاً من عدة عبارات if/if/if/else كما يلي:

```
switch (var) {

        case value1: … break;

        case value2: … break;

        …..

        default: …. break;

}
```

يتم تقييم قيمة المتحول var و مقارنتها مع كل قيمة معطاة عند حصول تطابق في القيمة يتم تنفيذ الأوامر ضمن سطر القيمة ثم يتم تنفيذ break للخروج من العبارة و إلا سيتم تنفيذ الأوامر لجميع القيم التي تليها.

الحلقات التكرارية

ثلاثة أنواع للحلقات في جافا:

1. حلقة while تختبر الشرط ثم تنفذ مجموعة من التعليمات ما دام الشرط صحيح و تكرر التنفيذ حتى يصبح الشرط غير صحيح.
2. حلقة do…while تشبه حلقة while لكنها تقوم بالتنفيذ أولاً ثم اختبار الشرط.
3. حلقة for تنفذ مجموعة من التعليمات عدد محدد من المرات, يتم تعريفها باستخدام قيمة بدائية + شرط توقف + تعليمة تغيير قيمة المتحول المستخدم ضمن الحلقة.

توجد نسخة محسنة من حلقة for تستخدم لكي يتم عبور عناصر مصفوفة كما يلي: for (int x:arrx){} حيث أن arrx مصفوفة من ints.

ضمن الحلقات يمكن استخدام العبارتين break لإيقاف الحلقة قسرياً و continue لإيقاف التكرار التالي قسرياً و الإنتقال للتكرار الذي بعده.

الصفوف

تستخدم الصفوف كقالب يتم من خلاله تعريف الكائنات, يمتلك الصف حالة state يتم تعريفها من خلال قيم الواصفات المعرفة ضمن الصف و سلوك behavior يتم تعريفه من خلال الطرائق methods.

غالباً ما تكون الواصفات خاصة بالصف private, و يتم استخدام طرائق عامة public للوصول إليها و تعديل قيمتها هذا ما يسمى بالتغليف encapsulation.

كل صف يمتلك باني constructor كي يتم ضمنه تهيئة واصفات الصف بقيم معينة ثابتة أو مدخلة من المستخدم.

ضمن ملف جافا الواحد نستطيع تعريف صف واحد عام له نفس إسم الملف بعد إزالة اللاحقة java. و عدد غير محدود من الصفوف الخاصة التي يمكن استخدامها ضمن الصف العام فقط.

أي ملف جافا يبدأ بتحديد اسم الحزمة package الموجود ضمنها الملف في حال كان موجود ضمن حزمة و يليه تعليمات import لصفوف و حزم أخرى مستخدمة ضمن الكود.

```
class Name {

        Name() {…} // constructor

        private int x; // A private attribute

        public void setX(int x){…} // public method

}
```

تمتلك أي طريقة معرفة ضمن الصف قيمة معادة + مجموعة وسطاء لها و جسم الطريقة الذي يتضمن التعليمات الخاصة بها.

إذا كان لطريقتين نفس الإسم و القيمة المعادة لكن مع وسطاء مختلفين نسمي هذا الأمر overloading.

إذا كان لطريقتين نفس الإسم و نفس القيمة المعادة و الوسطاء نسمي هذا الأمر override, نلاحظ هذه الحالة ضمن الصفوف الأبناء عندما يحقق صف إبن طريقة موجودة ضمن الصف الأب لكن بشكل مختلف.

الكلمة this ضمن أي طريقة تشير إلى الكائن الحالي.

يمكن تعريف طريقة بعدد غير محدود من الوسطاء من نوع معين كما يلي var ..type, شريطة أن يكون التعريف في نهاية قائمة وسطاء الطريقة.

الطريقة finalize يتم إستدعاؤها قبل تدمير الكائن من قبل ال destructor.

الوراثة

يتم استخدام الوراثة عندما يتشارك عدد من الصفوف ببعض الواصفات و الطرائق المشتركة, فبدلاً من تعريف عدة صفوف و كل واحد منها يتضمن نسخة من هذه الطرائق المشتركة يتم تعريف صف واحد يتضمن هذه الطرائق المشتركة و باقي الصفوف ترث هذه الطرائق و الواصفات منه.

ضمن كائنات الصف الإبن نستطيع استخدام التابع ()super للوصول لكائن الصف الأب, هذا الأمر ضروري خصوصاً ضمن الباني لاستدعاء باني الصف الأب من الصف الإبن و نهيئة الواصفات المشتركة.

نقول أن كائنات الصف الإبن ترتبط بعلاقة IS-A مع كائنات الصف الأب, يمكن أن ترتبط الكائنات أيضاً بعلاقة HAS-A عندما تكون واصفات صف ما هي من صف آخر, هذه العلاقات ستتوضح شمن الأمثلة.

تمرين

قم لكتابة برنامج جافا يصف العلاقة بين الأشكال بحيث يكون لدينا صف مجرد اسمه Shape يمتلك الواصفة التالية: sidesNum أي عدد الجوانب للشكل و الطرائق ()draw لرسم الشكل (يقوم بطباعة رسالة تشير إلى أن الشكل قد رسم), ()contour لحساب محيط الشكل و ()area لحساب مساحته.

جميع هذه الطرائق هي مجردة سيتم تحقيقها في الصفوف الأبناء.

الصف Rectangle يرث الصف السابق و يمثل مستطيل, تكون فيه قيمة sidesNum هي أربعة غير قابلة للتغيير, و يعرف واصفتين إضافيتين هما length و width يقوم بتحقيق الطرائق السابقة.

الصف Square يرث من المستطيل و يكون فيه length = height دوماً يحقق الطريقة ()draw فقط.

الصف Triangle يمثل مثلث و يرث الصف Shape, يكون فيه sidesNum = 3 غير قابلة للتغيير و يعرف الواصفات الإضافية side1 و side2 و side3 و type الذي يمثل enum لأنواع المثلثات, يحقق جميع الطرائق السابقة.

الصف Circle يمثل دائرة و يرث من Shape, تكون فيه sidesNum = 0 غير قابلة للتغيير و يعرف الواصفة الإضافية diameter كما يحقق الطرائق السابقة.

حل التمرين موجود على github على هذا الرابط https://github.com/mohsenSy/java-1

وظيفة

قم بكتابة برنامج جافا يتضمن الصفوف التالية:

الصف Human الذي يمثل أي إنسان و هو صف مجرد يتضمن الواصفات age, gender, length, job, name و الطرئق ()eat و ()sleep غير محققة ضمن الصف.

الصف Teacher الذي يصف مدرس, تكون فيه قيمة "job = 'teacher لا تغير و يتضمن أيضاً degree أي الشهادة التي حصل عليها المدرس و courses التي تمثل مصفوفة المناهج التي يدرسها (سيتم تعريف صف يمثل كل منهج course).

الصف Student يرث من Human و تكون قيمة "job = "student لا تغير و يتضمن سنة الدراسة (سيتم تعريف صف بمثل السنة أيضاً) و بالإضافة لمصفوفة المناهج التي يدرسها.

الصف Year يمثل سنة دراسية و يمتلك واصفة courses للمواد المعرفة ضمن السنة.

الصف Course يمثل منهاج ضمن سنة دراسية و يمتلك هذه الواصفات: year, name, teacher and practicalMark.