psybergate

Language Overview (Vac Work)

Project: java_basics

Topic Package: langoverview

General

- Java is an OO language (based on the principles of OO)
- Governed by the Java Language Specification – this is a formal document
- How does it execute:
    - Main method is entry point

Key Java Members

- The components that make up a Java class is referred to as the ***Java Members***
- The 5 key members:
  - Class/Interface
  - Method
  - Variable
  - Constructor
  - Parameter
- Others include: package / enum / local variable (see java.lang.annotation.ElementType for a list of the Java Members)

psybergate

Class

- Class is the general code artefact:
  - Package namespace
  - Many classes per file, but only one public class, which also corresponds with the name of the file

Types

- Primitive types
  - byte, short, int, long, float, double, boolean, char
  - Array []
- Class types
  - User defined (you can define anything as a type)
  - JDK Libraries / Other Libraries
  - Common class types:
    Date / String / Array (not visible) / Wrapper / BigDecimal / BigInteger
- Interface types
  - Only applicable to Object References, and not Objects
- Note : Class and Interface Types are collectively referred to a ***Reference Types***

Object

- Object is an instance of a Class:
  - Typically the “new” operator is used to create an object

```
Customer c = new Customer();
```

  - c is the Object Reference (which can be of type Class / Interface)
  - new Customer() is an actual Object – it must be an instance of a Concrete Class.

Method

- Methods define behaviours on classes / objects:
  - Method Signature (name and arguments – see JLS 8 Section 8.4.2 pg 232) – simply, the signature is the name and the parameter list
  - Instance/Object variable versus static variable
  - Static initialisers
  - Object/Instance initialisers

Variables

- Static
- Instance / Object
- Local

Packages / Imports

- Package – namespace
- Normal imports
- Static Imports (since Java 5)

Identifiers

- Java has some rules as to how various members are named (e.g. class/methods/variables cannot have a “-” in them)

Naming/Coding Conventions

- Follows Camel Case / Pascal Case notation:
  - Class names have first letter of each word capitalized – this is Pascal Case (which is a subset of Camel Case)
  - Method names has first letter of first word in lowercase, and there after first letter of each word in upper case
  - Variable names – same as method names
  - Other: constants, tests

Coding Standards

- Generally a good idea to setup coding standards
- See java-basics:JavaConstructCodeExamples
- IDE formatting standards and other tools such as CheckStyle can be used to manage / enforce some of these standards.

psybergate

Operators

- +, -, *, /, %
- = (assignment)
- ! (not)
- != (not equal)
- % (mod)
- ++, --
- += , -=, *=, /=
- &&, | |, &, |
- ==
- >, >=, <, <=
- …
- etc

Literals

- Literals are valid “hard-coded values” for a type – e.g. 1 is a literal for an int.
  - Examples:

```
String s = “abc”;
int i = 1;
boolean b = true;
Integer zero = 0;
```

java.lang.Object

- God class in Java
- All classes extend Object
- Why do you think Object is useful?
- Show key methods:
    - equals()
    - hashCode()
    - toString()
    - etc.

psybergate

Conditional Statements

- if
- if/else
- if/else if
- switch

psybergate

Loops

- for (and since Java 5 : enhanced-for loop, also known as the foreach statement)
- do
- while
- How do u write an infinite “for” loop?

```
for(; ;) {
}
```

- Discuss break and continue

Comments

- Single line
- Block comments
- Javadoc comments

Modifiers

- Modifies the behaviour (in some way) of a Java member (class/method/variable/parameter/Constructor/ etc):
  - static
  - final (see homework)
  - abstract
  - etc.

Scope/Access Modifiers

- Specifies scope of a Java member (class/method/variable):
  - private
  - package (friendly/default)
  - protected
  - public

Wrapper Classes

- Each primitive type in Java has its own Wrapper Class:
  - Byte
  - Short
  - Integer
  - Long
  - Float
  - Double
  - Boolean
  - Character

Purpose of Wrapper Classes

- Primarily introduced so that primitives can be wrapped in Objects
- Particularly useful (and very necessary) for Collections
  - See List#add() method

psybergate

Other

- Exceptions
- Enums
- Interfaces
- Arrays
- Collections
- Annotations
- Generics
- Lambdas
- …

psybergate

Homework 1a – hw1a

- Get a feel for what is in the Java Language Specification
- Explore the size of byte, short, int, long in code, in terms of min and max values
- Create a .java file with more than 1 class, and note your observations
- Find the source code for the following classes (in your IDE):
  - NullPointerException
  - String
  - Date

Homework 2a – hw2a

- When do static variables and static initialisers execute (see if you can write code that illustrates this)
  - Also, write an static initialiser that throws a new RuntimeException() and note your observations
- When do instance initialisers execute (see if you can write code to illustrate this)
- Anyone knows the difference between an object variable and a local variable (in terms of what modifiers you can use)? Again, write code to show this

Homework 3a - hw3a

- Attempt to setup your IDE to mimic our coding standards (or at least most of it):
  - Theme: Darcula
  - Font : Fira Code (non-proportional font)
  - Use spaces only for indents (not tabs)
  - Indents = 2 spaces
  - Every object/static variable on a separate line with one space between object/static variables – even the very first object/static variable
  - Opening brace { on same line
  - One space on either side of every operator
  - Maximum line width 100
- Make sure your IDE is correctly setup with these standards, so we can work effectively together

Homework 4a – hw4a : Operators

- Write sample code that illustrates the following operators:
  - % (mod)
  - ++ (both pre and post)
  - == (with primitives and Object references)
  - && and & (clearly illustrating the differences)
  - || and | (clearly illustrating the differences)
  - +=
  - switch

Homework 5a – hw5a: access modifiers

- Write code the clear illustrates the usage of the scope modifiers:
    - private
    - package (default or friendly)
    - protected
    - public

psybergate

Homework 6a – hw6a: static modifier

- Write code that shows clearly the difference between using static and not using static:
  - in a variable declaration;
  - in a Constructor declaration;
  - in a method declaration
  - in a class declaration;

Homework 7a – hw7a: final modifier

- Write code that shows the various usages of the final modifier:
  - in a class
  - in a method
  - in a constructor
  - in a public static final (variables and methods)
  - in a private final (variables and methods)
  - In a argument/parameter
- Note : Make sure you clearly understand this operator – it is probably the most overloaded Java modifier

Homework 8a – hw8a (Naming Conventions)

- See https://www.oreilly.com/library/view/java-8-pocket/9781491901083/ch01.html

- Naming is very important, so do more reading on Class and Interface names as they form the core of OO abstraction. In particular interface names can be harder to understand – see http://wiki.c2.com/?InterfacesShouldBeAdjectives

Homework 9a – hw9a : Java Coding Standards / Guidelines

- See class JavaCodingStandardsAndGuidelines2019 (don’t worry about this)

Homework 10a – hw10a : Floating point numbers

- https://introcs.cs.princeton.edu/java/91float/