Introducing Java 8

Java 8

- "The biggest change to Java since its inception"
- Many new language features
  - Default and Static Interface Methods
  - Lambda expressions
  - Method references
- New libraries
  - Streams
  - Date/time
  - Concurrency updates
- JVM changes
  - PermGen disappears

© J&G Services Ltd, 2017

Default Interface Methods

- Pre-Java8 interfaces only contain
  - constants
  - method signatures
- Java8 Interfaces can contain default methods
  - methods marked by keyword **default**
  - with a method body

```
interface InterfaceA {
        public void saySomething();
        default public void sayHi() {
                System.out.println("Say Hi");
        }
}

class MyClass implements InterfaceA {
        @Override
        public void saySomething() {
                System.out.println("Hello World");
        }
}
```

© J&G Services Ltd, 2017

Default Interface Methods

- Multiple interfaces can have the same default methods
  - need to override in implementing class

```
interface InterfaceC {
    public void saySomething();
    default public void sayHi() {
        System.out.println("Hi from InterfaceC");
    }
}
interface InterfaceD {
    default public void sayHi() {
        System.out.println("Hi from InterfaceD");
    }
}
class MyClass3 implements InterfaceC, InterfaceD {
    public void saySomething() {
              System.out.println("Say Something");
    }
    public void sayHi() {
        System.out.println("sayHi() in MyClass3");
    }
}
```

© J&G Services Ltd, 2017

Default Interface Methods

- Can call interface method if required
    - need to prefix call to super with type

```
class MyClass4 implements InterfaceC, InterfaceD {

    @Override
    public void saySomething() {
        System.out.println("Hello World");
    }

    // If want to call specific method in interface need to
    // specify the type before super
    @Override
    public void sayHi() {
       InterfaceC.super.sayHi();
    }

}
```

© J&G Services Ltd, 2017

Static Interface Methods

- Can now define static methods in interfaces
    - linked to type not to instance
    - in this case the interface

```
public interface InterfaceWithStatic {
       int getSomething();
       static int getSomethingStatic() {
              return 42;
       }
}
```

```
class SomeClass implements InterfaceWithStatic {
       public int getSomething() {
              return 0;
       }
}
```

© J&G Services Ltd, 2017

Static Interface Methods

- Can now define static methods in interfaces
  - linked to type not to instance
  - in this case the interface

```
public class StaticMethodExample {
    public static void main(String[] args) {
        // Can call static method directly on interface
        System.out.println(
            InterfaceWithStatic.getSomethingStatic());

        // Can't call static interface method on
        // implementing type
        //  System.out.println(
        //      SomeClass.getSomethingStatic());

        SomeClass s = new SomeClass();
        System.out.println(s.getSomething());
        // Can't call static interface method on object
       // System.out.println(s.getSomethingStatic());
       }
}
```

© J&G Services Ltd, 2017

Default & Static Interface Methods

- Interface default methods
  - helps to avoid utility classes
  - helps to extend interfaces without breaking existing code

- Interface static methods
  - part of interface not implementing class
  - can't be overridden by implementing class
  - may reduce the need for abstract classes

© J&G Services Ltd, 2017

Streams

- Java8 introduces a Streams API
- Supports function-style operations on streams of elements
- Integrated into the Collections API
- Can apply bulk operations to contents of collections etc.

```
List<Item> shoppingList = new ArrayList<>();
...
System.out.println(
    shoppingList.stream()
                .map(it -> it.getPrice())
                .max(Integer::max));
```

© J&G Services Ltd, 2017

New Date Time API

- Addresses many of the issues associated with old API
  - `java.util.Date` is not thread safe
  - `java.sql.Date toInstant()` is unsupported
  - variety of defaults used for year, month and day
  - difficulty in time zone handling
- New API in `java.time` package
- Local
  - a simplified date time API avoiding time zone issues
- Zoned
  - specialized date time API for time zones

© J&G Services Ltd, 2017

Concurrency Updates

- `CompletableFutures` added
- Class `CompletableFuture<T>`
  - A Future that may be explicitly completed
- Added support for Stream processing
  - parallel streams
- Additional concurrency classes
  - e.g. java.util.concurrent.ConcurrentHashMap class
- New classes in `java.util.concurrent.atomic`
  - for maintaining a single count or sum etc.

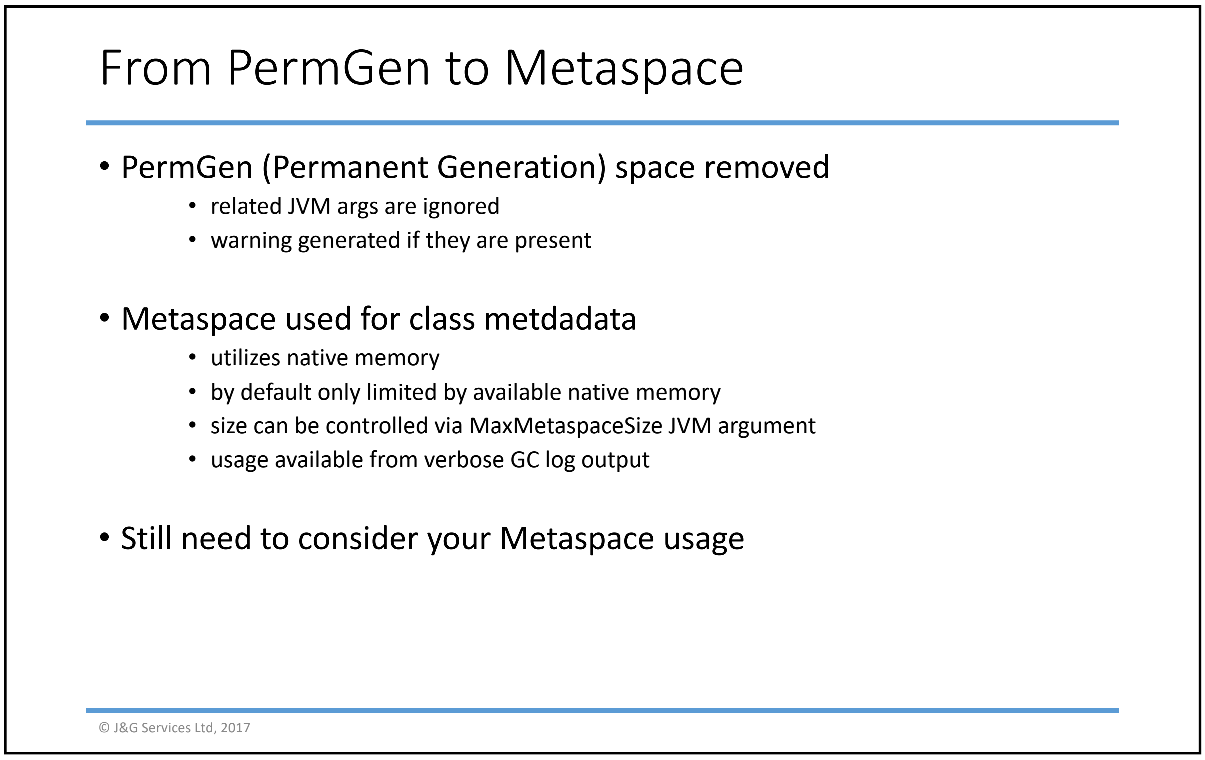
© J&G Services Ltd, 2017

Annotation Changes

- Type annotations
  - can now define annotations used on runtime values
  - an example is provided by the Chekcer framework which has
  - `@Nullable` – can be `null`
  - `@NonNull` – indicates non `null` value required
  - `@ReadOnly` – ensures an immutable object
- Repeating Annotations
  - can apply same annotation multiple times
  - defined using `@Repeatable`

```
@Alert(role="Manager")
@Alert(role="Admin")
public class UnauthorizedAccessException ...
```

© J&G Services Ltd, 2017



From PermGen to Metaspace

- PermGen (Permanent Generation) space removed
    - related JVM args are ignored
    - warning generated if they are present
- Metaspace used for class metdadata
    - utilizes native memory
    - by default only limited by available native memory
    - size can be controlled via MaxMetaspaceSize JVM argument
    - usage available from verbose GC log output
- Still need to consider your Metaspace usage

© J&G Services Ltd, 2017