CS2030S

Programming Methodology II

Lab 04

Lab 04 Preparation

Lab 04 Preparation

Factory
- Private

Factory Method

Private Constructor

> Write a class **A** that behaves as follows:

```
jshell> new A()
|  Error:
|  A() has private access in A
|  new A()
|  ^-----^
jshell> A.construct()
$4 ==> A@26be92ad
jshell> A.construct()
$5 ==> A@224edc67
```

[#]The aim is to helps you understand the initial `Probably<T>`

Lab 04 Preparation

Factory
- *Private*
- *Sharing*

Factory Method

Shared Object

> Modify `construct` method of A so that it always return the same instance:

```
jshell> A.construct()
$4 ==> A@26be92ad
jshell> A.construct()
$4 ==> A@26be92ad
```

[#] This is the singleton pattern

Lab 04 Preparation

Factory
- *Private*
- *Sharing*
- *Parameterized*

Factory Method

Shared Object + Parameterized Constructor

> Modify `construct` method of `A` so that it takes in an `int` and always return the same instance when the argument is `0`:

```
jshell> A.construct(0)
$4 ==> A@26be92ad
jshell> A.construct(0)
$4 ==> A@26be92ad
```

```
jshell> A.construct(1)
$5 ==> A@224edc67
jshell> A.construct(1)
$4 ==> A@4b9e13df
```

[#]This is an example of caching. Using codes from lecture, we may want to cache `Point(0, 0)`.

Lab 04 Preparation

Factory Method

Code for A

```
class Probably<T> {
  private int x;
  private static A zero = new A(0);
  private A(int x) {
    this.x = x;
  }
  public static A construct(int x) {
    if (x == 0) {
      return zero;
    }
    return new A(x);
  }
}
```

Lab 04 Preparation

Factory
Probably?

```
class Probably<T> {
  private final T value;
  private static final Probably<?> NONE = new Probably<>(null);
  private Probably(T value) {
    this.value = value;
  }
  public static <T> Probably<T> none() { // factory 1
    @SuppressWarnings("unchecked")
    Probably<T> res = (Probably<T>) NONE;
    return res;
  }
  public static <T> Probably<T> just(T value) { // factory 2
    if (value == null) {
      return none();
    }
    return (Probably<T>) new Probably<>(value);
  }
}
```

Lab 04 Preparation

Factory
Probably?
- Immutable

Probably

Immutable Class

1. The fields are all `final`
2. No setter
3. No getter

Lab 04 Preparation

Factory
Probably?
- Immutable

Probably

Immutable Class

1. The fields are all `final`
2. No setter
3. No getter

Question

What happen if we want to change the value of the field?

Lab 04 Preparation

Factory
Probably?
- *Immutable*
- *Comparison*

Probably

Generic Comparison

> Remember, you are not allowed to use raw types anymore! *(wildcards to the rescue)*

Raw Types

```
if (obj instanceof Probably) {
  :
}
```

Wildcards

```
if (obj instanceof Probably<?>) {
  :
}
```

Lab 04 Concepts

Lab 04 Concepts

PECS
- Actionable

Producer Extends, Consumer Super

PECS on Actionable

> Consider a method `void foo(P p)` where:
>
> - `P` is the type of the parameter

Is the parameter:

1. produced a value to be used by the method `foo`?
2. consumes the value generated by the method `foo`?

Lab 04 Concepts

PECS
- *Actionable*
- *Immutatorable*

Producer Extends, Consumer Super

PECS on Immutatorable

> Consider a method `R foo(P p)` where:
>
> - `R` is the return type
> - `P` is the type of parameter

Which of the following is the producer and which of the following is the consumer?

1. The parameter *(of type `P`)*
2. The return value *(of type `R`)*

Lab 04 Concepts

PECS
- *Actionable*
- *Immutatorable*
- *Applicable*

Producer Extends, Consumer Super

PECS on Applicable

> Consider a method `R foo(C<P> p)` where:
>
> - `R` is the return type
> - `C<P>` is the *(complex)* type of parameter

Determine the following validity of the following statement:

- We should accept the superclass of `P`.
- We should accept the subclass of `P`.

[#]Please refer to the example usage

Lab 04 Concepts

PECS
Notes

Notes on Lab 4

- Please understand the requirement of Lab 4
- Avoid raw types
- Use `@SuppressWarnings` responsibly

Looking Ahead

Looking Ahead

Panopto

Panopto Recording

> Please be familiar with Panopto recording. Panopo will be used for recording for:
>
> 1. Midterm
> 2. PE1
> 3. PE2

Looking Back

Looking Back

Lab 3

Lab 3 Interesting Solution

```
jshell> /exit
|    Goodbye
```