Peer Assisted Study Session

FIT2099 - Week 9

Monash University

Objectives

- Explore enumerations
- Explain Connascence

Estimated Time

Question 1 (5 Minutes)
Question 2 - Question 3 (8 Minutes)
Question 4 (5 Minutes)
Question 5 (5 Minutes)
Question 6 (10 Minutes)
Question 7 (5 Minutes)

Questions

1. Define an enumeration class Direction with the constants UP, DOWN, LEFT, RIGHT

2. Define a instance method turn that rotates a direction one step clockwise (e.g. UP becomes RIGHT)

3. Define an instance method getOpposite that returns the opposite direction

4. Define a static method oppositeOf that returns the opposite of its argument

5. Define a static method randomDirection that returns a random direction value

6. Generalize the randomDirection method by defining it elsewhere (perhaps the Main class or any other driver class) so that it works for any enumeration type.

For example, given an enum type Foo, one should be able to use it as such: Foo randomFoo = Main.randomDirection(Foo.class)

```
import java.util.*;

enum Direction {
UP,
DOWN,
LEFT,
RIGHT;

private Direction d;

public Direction rotateClockwise(){
  if (d == Direction.UP){
    return Direction.RIGHT;
  } else if (d == Direction.RIGHT){
    return Direction.DOWN;
  } else if (d == Direction.DOWN){
    return Direction.LEFT;
  }
  return Direction.UP;
}

public Direction getOpposite(){
    if (d == Direction.UP){
      return Direction.DOWN;
    } else if (d == Direction.RIGHT){
      return Direction.LEFT;
    } else if (d == Direction.DOWN){
      return Direction.UP;
    }
    return Direction.RIGHT;
  }

public static Direction oppositeOf(Direction d){
    if (d == Direction.UP){
```

```
      return Direction.DOWN;
    } else if (d == Direction.RIGHT){
      return Direction.LEFT;
    } else if (d == Direction.DOWN){
      return Direction.UP;
    }
    return Direction.RIGHT;
  }

public static Direction randomDirection(Direction d){
  Random rand = new Random();
  Direction[] values = d.getDeclaringClass().getEnumConstants();
  return values[rand.nextInt(values.length)];
  }


}

class Enums{
  private static Random rand = new Random();

  public static <T extends Enum<T>> T random(Class<T> ec){
    return getRandom(ec.getEnumConstants());
  }

  public static <T> T getRandom(T[] values){
    return values[rand.nextInt(values.length)];
  }
}


enum Activity{
SITTING,
LYING,
STANDING,
HOPPPING,
RUNNING,
```

```
DODGING,
JUMPING,
FALLING,
FLYING;
}



class Main {
public static void main(String[] args) {
  Direction d = Direction.LEFT;

  // Instance Methods
  System.out.println("\nRotate Clockwise");
  System.out.println(d.rotateClockwise());
  System.out.println("\nGet Opposite");
  System.out.println(d.getOpposite());

  // Static Methods
  System.out.println("\nStatic Opposite Of");
  System.out.println(Direction.oppositeOf(Direction.LEFT));
  System.out.println("\nStatic Random Direction");
  System.out.println(Direction.randomDirection(Direction.UP));

  System.out.println("\nGet Random Direction");
  for(int i = 0; i<20;i++){
    System.out.print(Enums.random(Direction.class) + " ");
  }

  System.out.println("\n\nGet Random Activities");
  for(int i = 0; i<20;i++){
    System.out.print(Enums.random(Activity.class) + " ");
  }



}
```

}

7. What is Connascence? Explain using the two main types.

A way to describe/measure dependencies in OO systems.

Static
Obvious from code structure
Can be automatically identified by IDE/analysis tools

Dynamic
Only obvious from close inspection/execution o Can't be easily identified by IDE
Generally, more concerning.