웰컴저축은행 Java Programming

메소드 추출 (Extract Method)

하나의 메소드가 너무 긴 경우, 기존 메서드에서 묶을 수 있는 코드를 추출해 새로운 메서드를 작성한다.

Before

```
public class ExtractMethod {
	private String name;
	private Orders orders;

	void printOwing() {
		Enumeration order = orders.elements();
		double outstanding = 0.0;

		// 배너 출력
		System.out.println("*******************");
		System.out.println("***  고객 외상  ***");
		System.out.println("*******************");

		// 외상액 계산
		while(order.hasMoreElements()) {
			Order each = (Order) orders.nextElement();
			outstanding += each.getAmount();
		}

		// 세부 내역 출력
		System.out.println("고객명 : " + name);
		System.out.println("외상액 : " + outstanding);
	}

}
```

After

```
public class ExtractMethod {
	private String name;
	private Orders orders;

	void printOwing() {
		printBanner();

		printDetail(getOutstanding());
	}

	private double getOutstanding() {
		Enumeration order = orders.elements();
```

```
            double outstanding = 0.0;

            // 외상액 계산
            while(order.hasMoreElements()) {
                  Order each = (Order) orders.nextElement();
                  outstanding += each.getAmount();
            }
            return outstanding;
      }

      private void printDetail(double outstanding) {
            System.out.println("고객명 : " + name);
            System.out.println("외상액 : " + outstanding);
      }

      private void printBanner() {
            System.out.println("********************");
            System.out.println("***  고객 외상  ***");
            System.out.println("********************");
      }

}
```

Inline Method

메서드 기능이 너무 단순해서 메서드 이름만 봐도 너무 뻔할 때는 그 메서드의 기능을 호출하는 메서드에 넣고 그 메서드를 삭제한다.

Before

```
      private int numberOfLateDeliveries;

      int getRating() {
            return (moreThanFiveRateDeliveries() ? 2 : 1);
      }

      boolean moreThanFiveRateDeliveries() {
            return numberOfLateDeliveries > 5;
      }
```

After

```
      private int numberOfLateDeliveries;

      int getRating() {
            return (numberOfLateDeliveries > 5 ? 2 : 1);
      }
```

어서션 도입 (Introduce Assertion)

코드 속에 성립해야 할 조건이 있는 경우, 주석으로 "이런 조건을 만족해야 한다.(성립해야 한다.)"라고 남기지 말고 어서션을 도입한다.

Before

```
import java.util.Arrays;

// 정수 배열을 정렬하는 코드
public class IntArraySorter {
	private final int[] datas;

	public IntArraySorter(int[] data) {
		// this.datas = data;
		this.datas = new int[data.length];
		System.arraycopy(data, 0, datas, 0, data.length);
	}

	public void sort() {
		for (int x = 0; x < datas.length; x ++) {
			int m = x;
			for (int y = x + 1; y < datas.length; y ++) {
				if (datas[m] > datas[y]) {
					m = y;
				}
			}

			// 여기에서 datas[m]는 datas[x] ~ datas[datas.length - 1]의 최소값이어야 한다.
			int v = datas[m];
			datas[m] = datas[x];
			datas[x] = v;
			// 여기에서 datas[0] ~ datas[x + 1]은 이미 정렬되어 있어야 한다.
		}
	}


	public String toString() {
		return Arrays.toString(this.datas);
	}

}
```

⇒ 반드시 만족해야 하는 조건이라면 assert 구문을 이용해서 확인하도록 한다.

```
import java.util.Random;

public class IntArraySorterTest {
	private static final Random ran = new Random(1234);

	private static void execute(int length) {
```

```
            // length 길이의 배열을 생성하고 난수 데이터를 추가
            int[] data = new int[length];
            for (int i = 0; i < length; i ++) {
                  data[i] = ran.nextInt(data.length);
            }

            // 데이터를 출력
            IntArraySorter sorter = new IntArraySorter(data);
            System.out.println("정렬전 : " + sorter);

            // 정렬 결과를 출력
            sorter.sort();
            System.out.println("정렬후 : " + sorter);
            System.out.println();
      }

      public static void main(String[] args) {
            execute(10);
            execute(10);
            execute(10);
            execute(10);
            execute(10);
      }
}
```

assert 표현식; ⇒ 표현식의 결과가 false 이면, java.lang.AssertionError 예외 발생
assert 표현식 : 옵션; ⇒ 표현식의 결과가 false 이면, java.lang.AssertionError 예외 발생하고, 옵션이 실행

After

```
import java.util.Arrays;

// 정수 배열을 정렬하는 코드
public class IntArraySorter {
      private final int[] datas;

      public IntArraySorter(int[] data) {
            // this.datas = data;
            this.datas = new int[data.length];
            System.arraycopy(data, 0, datas, 0, data.length);
      }

      public void sort() {
            for (int x = 0; x < datas.length - 1; x ++) {
                  int m = x;

⇒ assert 동작 여부를 테스트할 경우 아래 코드를 주석 처리
//                for (int y = x + 1; y < datas.length; y ++) {
//                      if (datas[m] > datas[y]) {
//                            m = y;
//                      }
//                }
```

```
                assert isMin(m, x, datas.length - 1) : this + ",  x = " + x;
                int v = datas[m];
                datas[m] = datas[x];
                datas[x] = v;
                assert isSorted(0, x + 1);
            }
        }

        // 여기에서 datas[0] ~ datas[x + 1]은 이미 정렬되어 있어야 한다.
        private boolean isSorted(int start, int end) {
            for (int i = start; i < end; i ++) {
                if (this.datas[i] > this.datas[i+1])
                    return false;
            }
            return true;
        }

        // 여기에서 datas[m]는 datas[x] ~ datas[datas.length - 1]의 최소값이어야 한다.
        private boolean isMin(int pos, int start, int end) {
            for (int i = start; i <= end; i ++) {
                if (this.datas[pos] > this.datas[i])
                    return false;
            }
            return true;
        }

        public String toString() {
            return Arrays.toString(this.datas);
        }

}
```

복잡한 논리와 씨름하기 예제

```
public class Range {
    int begin;
    int end;

    public Range(int begin, int end) {
        this.begin = begin;
        this.end = end;
    }

    boolean overlapsWith(Range r) {
//      [0, 2)가 [2, 4)와 겹치는 것으로 계산
//          return (begin >= r.begin && begin <= r.end) ||
//                  (end >= r.begin && end <= r.end);

// begin/end가 r에 완전히 포함되는 경우가 계산되지 않음
//          return (begin >= r.begin && begin < r.end) ||
//                  (end > r.begin && end <= r.end);
```

```
            return (begin >= r.begin && begin < r.end) ||
                        (end > r.begin && end <= r.end) ||
                        (begin <= r.begin && end >= r.end);
      }

      public static void main(String[] args) {
            Range a = new Range(1, 3);
            Range b = new Range(3, 6);
            Range c = new Range(6, 8);
            Range d = new Range(2, 8);

            System.out.println(a.overlapsWith(b)); // false
            System.out.println(a.overlapsWith(c)); // false
            System.out.println(a.overlapsWith(d)); // true
            System.out.println();

            System.out.println(b.overlapsWith(a)); // false
            System.out.println(b.overlapsWith(c)); // false
            System.out.println(b.overlapsWith(d)); // true
            System.out.println();

            System.out.println(c.overlapsWith(a)); // false
            System.out.println(c.overlapsWith(b)); // false
            System.out.println(c.overlapsWith(d)); // true
            System.out.println();

            System.out.println(d.overlapsWith(a)); // true
            System.out.println(d.overlapsWith(b)); // true
            System.out.println(d.overlapsWith(c)); // true
            System.out.println();

      }
}
```

매직 넘버를 기호 상수로 치환 (Replace Magic Number with Symbolic Constant)

Before - 로봇에게 명령을 전달하는 코드

```
public class Robot {
      private final String name;

      public Robot(String name) {
            this.name = name;
      }

      public void order(int command) {
            if (command == 0) {
```

```
                System.out.println(name + " walks.");
            } else if (command == 1) {
                System.out.println(name + " stops.");
            } else if (command == 2) {
                System.out.println(name + " jumps.");
            } else {
                System.out.println("Command error. command = " + command);
            }
        }
}
```

```
public class Main {
        public static void main(String[] args) {
                Robot robot = new Robot("Andrew");
                robot.order(0); // walk
                robot.order(1); // stop
                robot.order(2); // jump
        }
}
```

After - 리팩토링

방법1. 매직 넘버를 기호 상수로 치환

```
public class Robot {
        public static final int COMMAND_WALK = 0;
        public static final int COMMAND_STOP = 1;
        public static final int COMMAND_JUMP = 2;

        private final String name;

        public Robot(String name) {
                this.name = name;
        }

        public void order(int command) {
                if (command == Robot.COMMAND_WALK) {
                        System.out.println(name + " walks.");
                } else if (command == Robot.COMMAND_STOP) {
                        System.out.println(name + " stops.");
                } else if (command == Robot.COMMAND_JUMP) {
                        System.out.println(name + " jumps.");
                } else {
                        System.out.println("Command error. command = " + command);
                }
        }
}
```

```
public class Main {
        public static void main(String[] args) {
```

```
		Robot robot = new Robot("Andrew");
		robot.order(Robot.COMMAND_WALK);
		robot.order(Robot.COMMAND_STOP);
		robot.order(Robot.COMMAND_JUMP);

		robot.order(4);
	}
}
```

방법2. 분류 코드를 클래스로 치환

```
public class RobotCommand {
	private final String name;

	public RobotCommand(String name) {
		this.name = name;
	}
}
```

```
public class Robot {
	public static final RobotCommand COMMAND_WALK = new RobotCommand("WALK");
	public static final RobotCommand COMMAND_STOP = new RobotCommand("STOP");
	public static final RobotCommand COMMAND_JUMP = new RobotCommand("JUMP");

	private final String name;

	public Robot(String name) {
		this.name = name;
	}

	public void order(RobotCommand command) {
		if (command == COMMAND_WALK) {
			System.out.println(name + " walks.");
		} else if (command == COMMAND_STOP) {
			System.out.println(name + " stops.");
		} else if (command == COMMAND_JUMP) {
			System.out.println(name + " jumps.");
		} else {
			System.out.println("Command error. command = " + command);
		}
	}
}
```

```
public class Main {
	public static void main(String[] args) {
		Robot robot = new Robot("Andrew");
		robot.order(Robot.COMMAND_WALK);
		robot.order(Robot.COMMAND_STOP);
		robot.order(Robot.COMMAND_JUMP);
	}
}
```

⇒ Main 클래스는 변경이 동일한 기능을 제공하나,

⇒ 기호 상수를 사용하지 않고 분류 코드를 직접 사용하면 컴파일 오류가 발생

방법3. 열거형(enum)으로 기호 상수 표현

enum : 서로 연관된 상수들의 집합 ⇒ https://opentutorials.org/module/1226/8025

```
public class Robot {
	public enum Command {
		WALK,
		STOP,
		JUMP
	};

	private final String name;

	public Robot(String name) {
		this.name = name;
	}

	public void order(Robot.Command command) {
		if (command == Command.WALK) {
			System.out.println(name + " walks.");
		} else if (command == Command.STOP) {
			System.out.println(name + " stops.");
		} else if (command == Command.JUMP) {
			System.out.println(name + " jumps.");
		} else {
			System.out.println("Command error. command = " + command);
		}
	}
}
```

```
public class Main {
	public static void main(String[] args) {
		Robot robot = new Robot("Andrew");
		robot.order(Robot.Command.WALK);
		robot.order(Robot.Command.STOP);
		robot.order(Robot.Command.JUMP);
	}
}
```

사용자 장소를 "도시, 나라" 포맷으로 출력하는 예제

Before - 한번에 여러 가지 작업을 수행

⇒ 코드를 이해하기도 어렵고, 새로운 요구사항을 반영하는 것도 어려움

```
	<script>
```

```
        var location_info = {
              "LocalityName" : "Santa Monica",
              "SubAdministrativeAreaName" : "",
              "AdministrativeAreaName" : "",
              "CountryName" : "USA"
        }

        function getLocationInfo() {
              var place = location_info["LocalityName"]; // e.g. "Santa Monica"
              if (!place) {
                  place = location_info["SubAdministrativeAreaName"]; // e.g. "Los Angeles"
              }
              if (!place) {
                  place = location_info["AdministrativeAreaName"]; // e.g. "California"
              }
              if (!place) {
                  place = "Middle-of-Nowhere";
              }
              if (location_info["CountryName"]) {
                  place += ", " + location_info["CountryName"]; // e.g. "USA"
              } else {
                  place += ", Planet Earth";
              }
              return place;
        }

        document.write(getLocationInfo());
    </script>
```

After 1 - 한번에 하나의 작업만 수행하도록 코드를 구성

⇒ 새로운 요구사항("나라가 USA인 경우, 나라 이름 대신 AdministrativeAreaName 값을 출력")을 쉽게 반영

```
    <script>
        var location_info = {
              "LocalityName" : "Santa Monica",
              "SubAdministrativeAreaName" : "",
              "AdministrativeAreaName" : "",
              "CountryName" : "USA"
        }

        function getLocationInfo() {
              /* location_info 값 읽기 */
              var town = location_info["LocalityName"]; // e.g. "Santa Monica"
              var city = location_info["SubAdministrativeAreaName"]; // e.g. "Los Angeles"
              var state = location_info["AdministrativeAreaName"]; // e.g. "CA"
              var country = location_info["CountryName"]; // e.g. "USA"

              /* 나라 값 계산 */
              var second_half = "Planet Earth";
              if (country) {
                  second_half = country;
              }
              if (state && country === "USA") {
                  second_half = state;
```

```
            }

            /* 도시 값 계산 */
            var first_half = "Middle-of-Nowhere";
            if (state && country !== "USA") {
                first_half = state;
            }
            if (city) {
                first_half = city;
            }
            if (town) {
                first_half = town;
            }

            /* 장소 값 계산 */
            return first_half + ", " + second_half;
        }

        document.write(getLocationInfo());
    </script>
```

After 2 - || 연산자를 이용하여 코드 개선

```
    <script>
        var location_info = {
            "LocalityName" : "Santa Monica",
            "SubAdministrativeAreaName" : "",
            "AdministrativeAreaName" : "",
            "CountryName" : "USA"
        }

        function getLocationInfo() {
            var town = location_info["LocalityName"]; // e.g. "Santa Monica"
            var city = location_info["SubAdministrativeAreaName"]; // e.g. "Los Angeles"
            var state = location_info["AdministrativeAreaName"]; // e.g. "CA"
            var country = location_info["CountryName"]; // e.g. "USA"

            var first_half, second_half;
            if (country === "USA") {
                first_half = town || city || "Middle-of-Nowhere";
                second_half = state || "USA";
            } else {
                first_half = town || city || state || "Middle-of-Nowhere";
                second_half = country || "Planet Earth";
            }

            return first_half + ", " + second_half;
        }

        document.write(getLocationInfo());
    </script>
```