CS 635 Advanced Object-Oriented Design & Programming
Fall Semester, 2021
Doc 22 Assignment 4 Comments
Dec 31, 2021

Copyright ©, All rights reserved. 2021 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.

```
public int process(PatternRequest request) {
    String inputString = request.getInputString();
    int indexToCheckFor = request.getIndex();

  if(indexToCheckFor < inputString.length() ) {
      char currentChar = request.getInputString().charAt(indexToCheckFor);

      if(_patternChar == currentChar) {
          request.setIndex(indexToCheckFor+1);

          if(_next != null){
              return _next.process(request);
          }else {
              return indexToCheckFor;
          }
      } else
          return -1;
    }else {
      return -1;
    }
  }
```

Notice how long it takes you to think through all the cases.

Once you have done that go to the next slide

```
public int process(PatternRequest request) {
    String inputString = request.getInputString();
    int indexToCheckFor = request.getIndex();

   if(indexToCheckFor >= inputString.length() ) {
            return -1;
         }
   char currentChar = request.getInputString().charAt(indexToCheckFor);

    if(_patternChar != currentChar) {
             return -1;
          }
    request.setIndex(indexToCheckFor+1);

     if(_next != null){
          return _next.process(request);
        }else {
           return indexToCheckFor;
        }
  }
```

Now how long it takes you to think through all the cases.

The first two cases end by returning right way so we don't have to look through all the cases to see how they end

How about using a constant NOT_FOUND for -1

RegularCharacterMatcher

```
def find_character_match(self, text, position):
    match_found = text[position] == self.character_to_match
    if match_found:
        if self.next_character_matcher:
            next_matched_position = self.next_character_matcher.find_character_match(text, position + 1)
            if next_matched_position == -1:
                return next_matched_position
            return position
        return position
    return -1
```

Python example of nested if statements

```
def find_character_match(self, text, position):
    if  text[position] != self.character_to_match:
         return NO_MATCH;
    if self.next_character_matcher:
        next_matched_position = self.next_character_matcher.find_character_match(text, position + 1)
        if next_matched_position == NO_MATCH:
            return NO_MATCH
     return position
```

Same functionality, simpler code

```
public interface PatternMatcher {
    void setNextChain(PatternMatcher next);
    boolean validate(String matchStr, int index);
}
```

What is being validated?

```
 public interface Handler<T> {
     void setNext(Handler<T> next);

     boolean handle(T request);
 }
```

What is being handled?

```
 public abstract class Matcher {
     protected Matcher nextMatcher;

     public abstract boolean match(String stringToMatch, int index);

     public void setNextMatcher(Matcher matcher) {
         this.nextMatcher = matcher;
     }
}
```

Match is a better name as it gives some indication of what being done

In CharacterMatchHandler

```
public int check(String inputString, int startIndex) {
        Integer currentCharIndex = -1;
        if (startIndex < 0 || startIndex >= inputString.length()) {
                return -1;
        }
        do {
                if (isStart) {
                        currentCharIndex = inputString.indexOf(currentCharacter, startIndex);
                } else {
                        if (inputString.charAt(startIndex) == currentCharacter.charValue()) {
                                currentCharIndex = startIndex;
                        } else {
                                currentCharIndex = -1;
                        }
                }
                if (currentCharIndex != -1) {
                        Integer chainCharIndex = checkNext(inputString, currentCharIndex + 1);
                        if (isStart && chainCharIndex.equals(-1)) {
                                startIndex = currentCharIndex + 1;
                        } else if (chainCharIndex.equals(-1)) {
                                return chainCharIndex;
                        } else if (chainCharIndex.equals(-2)) {
                                return currentCharIndex;
                        } else {
                                return currentCharIndex;
                        }
                } else {
                        return currentCharIndex;
                }
        } while (isStart && startIndex < inputString.length());

        return currentCharIndex;
}
```

This is complex.

What does -1 and -2 mean? Use constants

If you create a separate node to be the head node in the chain then CharacterMatchHandler becomes very simple

```
public abstract class MatchHandler {
	protected Character currentCharacter;
	protected boolean isStart;
	protected MatchHandler nextHandler;

	public MatchHandler(Character currentCharacter, boolean isStart) {
		super();
		this.currentCharacter = currentCharacter;
		this.isStart = isStart;
	}

	public Character getCurrentCharacter() {
		return currentCharacter;
	}

	public MatchHandler setNext(MatchHandler next) {
		this.nextHandler = next;
		return next;
	}

	protected int checkNext(String inputString, int startIndex) {
		if (nextHandler == null) {
			return -2;
		}
		return nextHandler.check(inputString, startIndex);
	}

	public abstract int check(String inputString, int startIndex);

}
```

Why is this needed?

An example of Template method, but the names need to be improved
What is the difference between checkNext and check?
Here it is fairly clear but a bit confusing in the subclasses