Regexp in Android and Java

Keishin Yokomaku @ Drivemode, Inc.
potatotips #22

@KeithYokoma

- Keishin Yokomaku at Drivemode, Inc.
- Work
  - Android apps
  - Android Training and its publication
- Like
  - Bicycle, Photography, Tumblr and Motorsport

Pattern.compile()

Android is not Java

- Android and Java have different implementation respectively on regexp API.
  - Same regexp, different result.
  - Write once, not run anywhere (☆ ‿ ☆)

What’s the matter?

Test env vs. runtime

- If your tests are running on JVM(e.g. with Robolectric)…
  - Some patterns pass the test, but won’t work at runtime.
  - Some patterns work at runtime, but won’t pass the test.

¯_(ツ)_/¯

Differences in detail

Supported flags

- Java
  - All flags defined at *Pattern* are supported.
- Android
  - Only CASE_INSENSITIVE, COMMENTS, DOTALL, LITERAL, MULTILINE, UNICODE_CASE, UNIX_LINES are supported.
  - If any other flags are set, RuntimeException will be thrown.

Android Pattern

```
public class Pattern {

  private Pattern(String pattern, int flags) throws PatternSyntaxException {
    if ((flags & CANON_EQ) != 0) {
      throw new UnsupportedOperationException(“CANON_EQ flag not supported”);
    }
    int supportedFlags = CASE_INSENSITIVE | COMMENTS | DOTALL |
        LITERAL | MULTILINE | UNICODE_CASE | UNIX_LINES;
    if ((flags & ~supportedFlags) != 0) {
      throw new IllegalArgumentException(“Unsupported flags: “ + (flags & ~supportedFlags));
    }
    this.pattern = pattern;
    this.flags = flags;
    compile();
  }
}
```

Android Pattern

```
public class Pattern {

  private Pattern(String pattern, int flags) throws PatternSyntaxException {
    if ((flags & CANON_EQ) != 0) {
      throw new UnsupportedOperationException("CANON_EQ flag not supported");
    }
    int supportedFlags = CASE_INSENSITIVE | COMMENTS | DOTALL |
        LITERAL | MULTILINE | UNICODE_CASE | UNIX_LINES;
    if ((flags & ~supportedFlags) != 0) {
      throw new IllegalArgumentException("Unsupported flags: " + (flags & ~supportedFlags));
    }
    this.pattern = pattern;
    this.flags = flags;
    compile();
  }
}
```

Android Pattern

```
public class Pattern {

  private Pattern(String pattern, int flags) throws PatternSyntaxException {
    if ((flags & CANON_EQ) != 0) {
      throw new UnsupportedOperationException("CANON_EQ flag not supported");
    }
    int supportedFlags = CASE_INSENSITIVE | COMMENTS | DOTALL |
        LITERAL | MULTILINE | UNICODE_CASE | UNIX_LINES;
    if ((flags & ~supportedFlags) != 0) {
      throw new IllegalArgumentException("Unsupported flags: " + (flags & ~supportedFlags));
    }
    this.pattern = pattern;
    this.flags = flags;
    compile();
  }
}
```

Java Pattern

```
public class Pattern {
  private void compile() {
    if (has(CANON_EQ) && !has(LITERAL)) {
      normalize();
    } else {
      normalizedPattern = pattern;
    }
    patternLength = normalizedPattern.length();

    // Copy pattern to int array for convenience
    // Use double zero to terminate pattern
    temp = new int[patternLength + 2];

    hasSupplementary = false;
    int c, count = 0;
    // Convert all chars into code points
    for (int x = 0; x < patternLength; x += Character.charCount(c)) {
      c = normalizedPattern.codePointAt(x);
      if (isSupplementary(c)) {
        hasSupplementary = true;
      }
      temp[count++] = c;
    }
    patternLength = count;   // patternLength now in code points

    // ……
  }
}
```

Character class

- Java
  - Matches only single byte characters
- Android
  - Matches both single byte and multi byte characters.
- Details here: http://bit.ly/1R73wkM

Regular expression engines

- Java
  - java.util.regex Engine
  - Conform Unicode Technical Standard #18 Level1 and Release 2.1”Canonical Equivalents”.
- Android
  - ICU(International Components for Unicode) Engine
  - Conform Unicode Technical Standard #18 Level 1 and Default Word Boundaries and Name Properties from Level2

Canonical Equivalents

- Canonically equivalent code point sequences are assumed to have the same appearance and meaning when printed or displayed.
  - e.g. “ü” and “ü” are canonically equivalent

Android is not Java ¯_(ツ)_/¯

Regexp in Android and Java

Keishin Yokomaku @ Drivemode, Inc.
potatotips #22