mid-term exam

과목명 : 모바일 소프트웨어(야간) - closed book
시험 일자 : 2019년 10월 22일(화)
시험 시간 : 18:00 ~ 19:00 (60min.)

1. (28점) **MyApplication 프로젝트에 대해 다음 질문에 답하시오.**

```
1   [redacted] edu.ourincheon.myapplication
2
3   [redacted] android.os.Bundle
4              androidx.appcompat.app.AppCompatActivity
5              kotlinx.android.synthetic.main.content_main.*
6
7   [redacted] MainActivity : AppCompatActivity() {
8
9       override [redacted] onCreate(savedInstanceState: Bundle?) {
10          super.onCreate(savedInstanceState)
11          [redacted](R.layout.activity_main)
12
13          textView.text = getString([redacted])
14      }
15  }
```

그림 1. MainActivity.kt

```
<resources>
    <string name="welcome_message">Hello World!</string>
    <string name="convert_string">Convert to Korean won</string>
```

그림 2. strings.xml(부분)

```
26      <Button
27          android:id="@+id/button"
28          android:layout_width=[redacted]
29          android:layout_height=[redacted]
30          android:layout_marginTop="64dp"
31          android:layout_marginBottom="32dp"
32          android:text=[redacted]
33          android:textAllCaps=[redacted]
34          android:textSize="20[redacted]"
```

그림 3. activity_main.xml(부분)

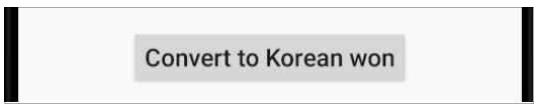


그림 4. AVD 실행 결과(부분)

가. (2점) 그림 1의 줄 1 빈칸을 채우시오. **package**
나. (2점) 그림 1의 줄 3 빈칸을 채우시오. **import**
다. (2점) 그림 1의 줄 7 빈칸을 채우시오. **class**
라. (2점) 그림 1의 줄 9 빈칸을 채우시오. **fun**
마. (4점) 그림 1의 줄 11 빈칸을 채우시오. **setContentView**
바. (6점) 그림 1의 줄 13 빈칸을 채우시오. 단, TextView에 출력할 내용은 strings.xml에 정의된 "Hello Word!" 문자열 리소스이다. **R.string.welcome_message**

※ **사)-자)는 그림 4의 AVD 실행 결과를 참고할 것**.

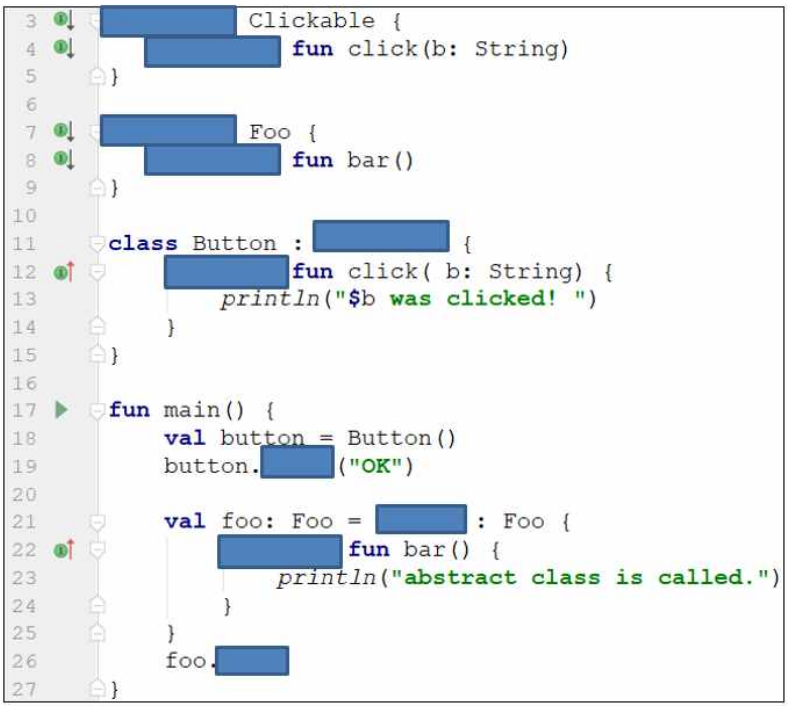
사. (2점) 그림 3의 줄 28 빈칸을 채우시오. **wrap_content**

아. (4점) 그림 3의 줄 32 빈칸을 채우시오. **wrap_content**

자. (2점) 그림 3의 줄 33 빈칸을 채우시오. **@string/convert_string**

차. (2점) 그림 3의 줄 34 빈칸을 채우시오. **sp**

2. (21점) 아래 kotlin 파일을 보고 다음 질문에 답하시오.



```
3   [          ] Clickable {
4   [          ]     fun click(b: String)
5   }
6
7   [          ] Foo {
8   [          ]     fun bar()
9   }
10
11  class Button : [          ] {
12      [          ] fun click( b: String) {
13          println("$b was clicked! ")
14      }
15  }
16
17  fun main() {
18      val button = Button()
19      button.[     ]("OK")
20
21      val foo: Foo = [     ] : Foo {
22          [          ] fun bar() {
23              println("abstract class is called.")
24          }
25      }
26      foo.[     ]
27  }
```

가. (3점) 줄 3의 빈칸을 채우시오. **interface**

나. (3점) 줄 4의 빈칸을 채우시오. **abstract**

다. (3점) 줄 11의 빈칸을 채우시오. **Clickable**

라. (3점) 줄 12의 빈칸을 채우시오. **override**

마. (2점) 줄 19의 빈칸을 채우시오. **click**

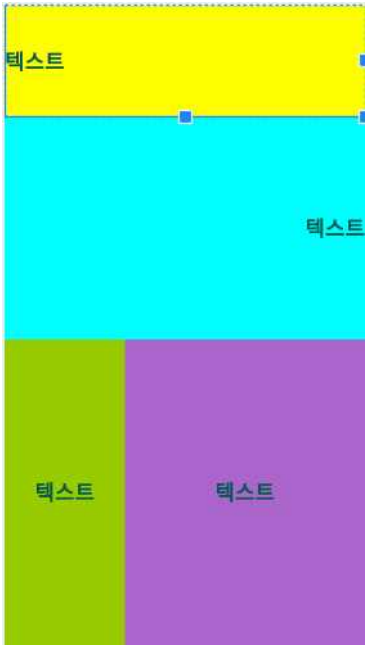
바. (3점) 줄 21의 빈칸을 채우시오. **object**

사. (2점) 줄 22의 빈칸을 채우시오. **override**

아. (2점) 줄 26의 빈칸을 채우시오. **bar( ) (괄호가 없으면 0점. 괄호 안에 공백 있어도 됨)**

3. (24점) 오른쪽 그림과 같은 화면을 만들려고 한다. 왼쪽은 레이아웃 XML 파일이다. 단, 문제와 관련 없는 부분은 삭제한 코드이다. root layout은 orientation 속성이 각기 다른 2개의 child LinearLayout을 갖고 있다. 각 child LinearLayout은 2개의 TextVIew를 child로 포함하고 있으며, orientation 속성에 따라 수평 또는 수직 방향으로 1:2 비율로 공간을 분배한다.

```
1 <LinearLayout>
2    <LinearLayout>
3        <TextView
4            android:layout_width="_________"
5            android:layout_height="_________"
6            android:layout_weight="_________"
7            android:gravity="_________|center_vertical"/>
8        <TextView
9            android:layout_width="_________"
10            android:layout_height="_________"
11            android:layout_weight="_________"
12            android:gravity="_________|center_vertical"/>
13    </LinearLayout>
14
15   <LinearLayout>
16        <TextView
17            android:layout_width="_________"
18            android:layout_height="_________"
19            android:layout_weight="_________"
20           android:gravity="_________" />
21        <TextView
22            android:layout_width="_________"
23            android:layout_height="_________"
24            android:layout_weight="_________"
25            android:gravity="_________"/>
26    </LinearLayout>
27 </LinearLayout>
```



가. (2점) 줄 5의 빈칸을 채우시오. **0dp**
나. (2점) 줄 6의 빈칸을 채우시오. **1**
다. (2점) 줄 7의 빈칸을 채우시오. **left 또는 start**
라. (2점) 줄 10의 빈칸을 채우시오. **0dp**
마. (2점) 줄 11의 빈칸을 채우시오. **2**
바. (2점) 줄 12의 빈칸을 채우시오. **right 또는 end**
사. (2점) 줄 17의 빈칸을 채우시오. **0dp**
아. (2점) 줄 19의 빈칸을 채우시오. **1**
자. (2점) 줄 20의 빈칸을 채우시오. **center**
차. (2점) 줄 22의 빈칸을 채우시오. **0dp**
카. (2점) 줄 24의 빈칸을 채우시오. **2**
타. (2점) 줄 25의 빈칸을 채우시오. **center**

4. (27점) 아래는 touch 이벤트를 처리하는 코드이다.

```
9       MainActivity : AppCompatActivity() {
10      override onCreate(savedInstanceState: Bundle?) {
11          super.onCreate(savedInstanceState)
12          (R.layout.activity_main)
13
14          activity_motion_event.setOnTouchListener(
15              :  {
16              fun onTouch(v: View?, e: MotionEvent?): Boolean {
17                  handleTouch(e)
18                  return true
19              }
20          })
21      }
22
23      private fun handleTouch(m:  ) {
24          if  return
25          val x =
26          val y =
27          var actionString = ""
28
29          when (m.action) {
30              MotionEvent.  -> actionString = "DOWN"
31              MotionEvent.  -> actionString = "UP"
32              MotionEvent.  -> actionString = "MOVE"
33              -> actionString = ""
34          }
35          textView.text = "$actionString : ($x, $y) "
36      }
37  }
```

가. (각 3점, 6점) 줄 15의 **빈칸 2곳을 모두** 채우시오. **object, View.OnTouchListener**

나. (3점) 줄 23의 빈칸을 채우시오. **MotionEvent?**

다. (3점) 줄 24의 빈칸을 채우시오. **(m == null)**

라. (3점) 줄 25의 빈칸을 채우시오. **m.x**

마. (3점) 줄 30의 빈칸을 채우시오. **ACTION_DOWN**

바. (3점) 줄 33의 빈칸을 채우시오. **else**

사. (6점) 줄 35는 소수점 이하 출력 제한이 없다. 소수점 이하 2째자리까지 출력하도록 줄 35의 코드를 수정하시오.

```
textView.text = "$actionString : (%.2f, %.2f) ".format(x, y)
```

아래와 같이 코딩하였을 경우, import java.text.DecimalFormat을 포함하지 않아도 정답 처리

```
val df = DecimalFormat("#.##")
df.roundingMode = RoundingMode.CEILING

textView.text = "$actionString : (${df.format(x)}, ${df.format(y)}"
```