Spring 2019 (Wednesday, March 13) Name: _____

CS 4330/5390: Mobile Application Development

Exam 1

This test has 9 questions and pages numbered 1 through 10.

Reminders

This test is closed-book. However, you are allowed to bring one page of notes (8.5 by 11 inches; both sides). Your notes must be your own, they must be hand written, and they must be turned in with your test.

This test is to be done individually, and you are not to exchange or share your notes with other students during the test.

If you need more space, use the back of a page. Note when you do that on the front.

This test is timed. Your test will not be graded if you try to take more than the time allowed. Therefore, before you begin, please take a moment to look over the entire test so that you can budget your time.

For program code, clarity is important; if your writing or code is sloppy or hard to read, you will lose points. Correct syntax also makes some difference.

There are 105 points all, including 5 bonus points.

---

1. (total 20 points) Short answers: multiple choice and fill-in-the-blank

   (a) (2 points) All Android apps have an automatically generated file named _____`.java`. It provides references, or resource Ids, to various resources used by the applications, e.g., layouts and images. That is, instead of hard coding such resources into an application, one externalizes and refers them by identifiers.

   (b) (2 points) A special view class named `android.view.`_____ can contain one or more other views. It is the base class of all layout views and provides the layout in which the contained views can appear and be ordered.

   (c) (2 points) This layout is a placeholder on screen that you can use to display typically a single view. Views that you add to it are always anchored to the top left of the layout. If you add multiple views, they are stacked.

      i. FrameLayout
      ii. GridLayout
      iii. LinearLayout
      iv. RelativeLayout

   (d) (2 points) A _____ represents a behavior or a portion of user interface in an activity. It is particularly useful in adapting a user experience across a wide range of devices, as they can be combined to build a multi-pane UI and be reused in multiple activities.

(e) (2 points) In general, you use the `startActivity()` framework method to invoke an activity. However, it doesn't return a result from the called activity to the calling activity. If want to receive data back from the called activity, you should instead use the _____() method to invoke it.

(f) (2 points) As stated in the previous question, an activity can indicate its intention to receive a result when invoking another activity. However, to actually receive data from the called activity, you override the _____() method in the calling activity class.

(g) (2 points) Android provides three different types of menus: *options* (*actions/tools*), *context*, and *popup*. Name the menu type that meets all the descriptions given below: _____

- It displays information related to a particular view of an activity.
- It is a floating menu owned by or associated with a view.
- Its menu items can be defined statically in an XML resource file or programmatically in source code.
- It can be activated by any UI event such as button click.

(h) (2 points) To provide the options (actions/tools) menu for your activity, you need to override two framework methods in your activity: `onCreateOptionsMenu()` and _____. The first method is called to populate a menu with menu items, and the second method is called when a menu item is selected.

(i) (2 points) To create a fragment, you must create a subclass of `android.app.Fragment` or its subclass, and then you have to override at least one framework method. Which of the following methods is most likely to be overridden in your fragment definition?

i. onAttach()
ii. onCreate()
iii. onCreateView()
iv. onActivityCreate()

(j) (2 points) The term _____ design refers to the adaptation of a layout design that fits an individual screen size and/or orientation. It is important to Android because it supports flexibility when designing an app that can work on multiple devices.

2. (4 points) One noticeable feature of Android programming is the use of XML. List at least four different uses of XML in Android programming.

3. (5 points) There are several different units of measurement that you can use to specify the size, or dimension, of an Android UI element. Which unit is recommended for specifying the size of a view? Justify your answer by stating the reason.

4. (5 points) The model-view-controller (MVC) design, or architecture, provides a way of separating user interface (UI) functionality. Describe briefly why its use in Android programming is more important than other UI programming like desktop or web programming.

5. (6 points) Describe the differences between *explicit* and *implicit* intents. Do not use the terms explicit and implicit in your description.

6. (10 points) There are several framework methods to be called on various stages of an activity from its creation to destruction.

   - Draw a diagram showing Android's activity lifecycle.
   - State the key difference between the HOME button and the BACK button in terms of activity lifecycle methods.

7. (total 25 points) You are to develop an app named CarShow that displays a list of cars. When a user clicks a model name, the app shows an image of the clicked car (see Figure 1 for sample screenshots). A model class named `Car` is provided, and each car has a model name and an image specified by an image resource identifier. You will be writing the CarShow app incrementally by completing provided skeletal code.

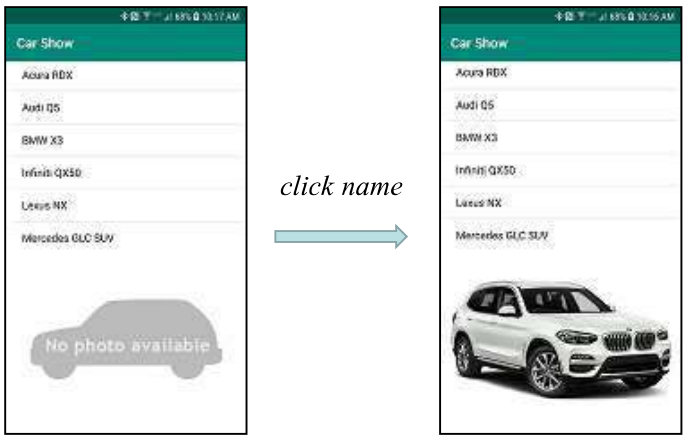

Figure 1: Sample screenshots

```
public class Car {

    /** Resource id of an image showing a text "No photo available". */
    public static final int NO_IMAGE = R.drawable.no_image;

    private static final List<Car> cars = Arrays.asList(
            new Car("Acura RDX", R.drawable.acura_rdx),
            // ...
            new Car("Mercedes GLC SUV", R.drawable.mercedes_glc_suv));

    /** Model name of this car. */
    public final String name;

    /** Resource id of an image of this car. */
    public final int imageId;

    public Car(String name, int imageId) {
        this.name = name;
        this.imageId = imageId;
    }

    public String toString() {
        return name;
    }

    /** Return all cars. */
    public static List<Car> cars() {
        return cars;
    }
}
```

(a) (2 points) What should be the image file name of a car, say Acura RDX, and where should the file reside? Assume that you use a single image for all devices regardless of their screen densities or resolutions. Be specific about the directory/file names.

(b) (8 points) Let us first define the UI of our app by completing the skeletal `activity_main.xml` file given below. As shown, the UI consists of a ListView and an ImageView. Let us give an equal height to these two views. For this, you may need to specify their attributes. *Do not change the top-level LinearLayout to another layout.*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    <!-- (a) WRITE YOUR CODE HERE IF NEEDED -->









    tools:context=".MainActivity">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        <!-- (b) WRITE YOUR CODE HERE IF NEEDED -->
```



```
        android:layout_margin="8dp"/>

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        <!-- (c) WRITE YOUR CODE HERE IF NEEDED -->
```



```
        android:layout_margin="8dp"/>
</LinearLayout>
```

(c) (15 points) Next, let us code the business logic of the app—showing a list of cars and displaying an image when a car is clicked. The app initially displays an image specified by the static field `Car.NO_IMAGE` (see the sample screenshots on page 5). The following APIs may be helpful in writing your code.

- `List<Car> Car.cars()`: static method to return all cars.
- **`void`** `ImageView.setImageResource(`**`int`**`)`: non-static method to set the image to be displayed. The image is specified by an image resource id.
- `ArrayAdapter(Context,` **`int`**`, List<T>)`: constructor to create an adapter from a list, e.g., **`new`** `ArrayAdapter<>(`**`this`**`, android.R.layout.simple_list_item_1, ... )`
- `T ArrayAdapter.getItem(`**`int`**`)`: non-static method to return an item at a specified position.
- `OnItemClickListener`: interface defined in `AdapterView` to handle click events.

```
public interface OnItemClickListener {
  public void onItemClick(AdapterView<?> parent, View view,
    int position, long id);
}
```

Hint: you don't need to define any other method.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // WRITE YOUR CODE HERE ...
    }
}
```

8. (total 20 points) You are to improve the user experience of the CarShow app by (a) providing a landscape mode-specific UI and (b) handling screen orientation changes (see Figure 2). Note that the same car image is displayed when the device's screen orientation changes.

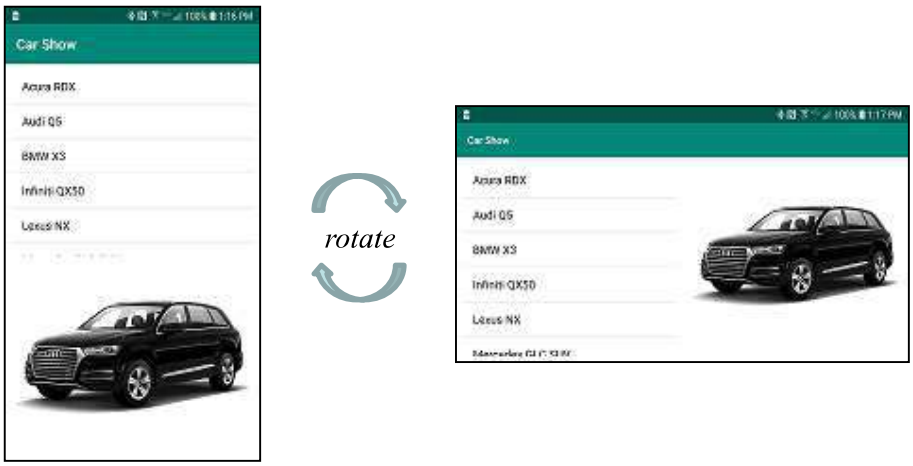

Figure 2: Landscape UI and screen orientation change

(a) (7 points) Write the full pathname (directory and file names) of the layout XML file for the landscape mode, and define the new layout by writing only the differences from the portrait mode.

(b) (13 points) Make changes to the `MainActivity` class to address screen orientation changes. That is, the same car image should be displayed upon screen orientation change. You may assume that the `Car` class now has the following method defined.

```
/** Return a car specified by its model name. */
public static @Nullable Car car(String mname) {
  return cars.stream().filter(car -> car.name.equals(mname)).findFirst().get();
}
```

You need to show only new code as well as changes to the existing code (if any), not the complete code. Hint: you may need to introduce one or two new methods depending on your implementation strategy.

9. (10 points) Here we are to provide a way for a user to view the webpage of a car. A car is now assumed to have an optional URL that specifies its webpage. For this, the `Car` class defines a new final field as shown below.

```
/** URL of this car's webpage, or null if no webpage is specified. */
public final @Nullable String url;
```

If a user clicks the image of a car, the app launches a (built-in) web browser to show the webpage of the car; if the car has no webpage, nothing happens (see Figure 3).

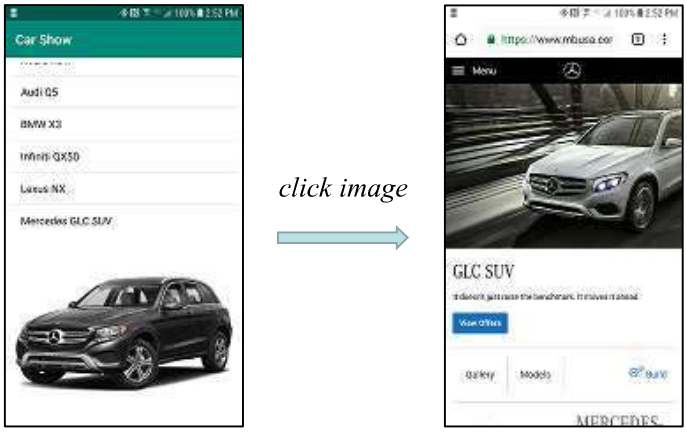

Figure 3: Viewing the webpage of a car

Show only new and changed code, not the complete code. Hint: how to launch a web browser? Use the `Uri.parse(String)` static method to convert a URL string to a URI object.