Private Test Runner

These are tests of private interfaces that can't easily happen in the regular flutter tests due to problems with test and implementation interdependence.

This gets around the problem of parts existing in more than one library by making a copy of the code under test.

The test script `bin/test_private.dart` tests private interfaces by copying the code under test into a temporary workspace. The test is then free to make the copied flutter source into a "part" of its own library by declaring a library and using the `part` directive with a relative path to include the parts. This way the test and the private interface are part of the same library, and the private interface can be accessed by the test.

The tests are run like so:

```
dart run bin/test_private.dart
```

One limitation is that the copied private API needs to be separable enough to be copied, so it needs to be in its own separate files.

To add a private test, add a manifest file of the form (assuming "my_private_test" is the name of the test) to the test subdir:

```
{
  "tests": [
    "my_private_test.dart"
  ],
  "pubspec": "my_private_test.pubspec.yaml",
  "deps": [
    "lib/src/subpackage/my_private_implementation.dart",
  ]
}
```

It will copy the files in `deps` relative to the `packages/flutter` directory into a similar relative path structure in the test temporary directory tree. It will copy the `pubspec` file into `pubspec.yaml` in the test temporary directory, and copy all of the `tests` into the top of the test temporary directory tree.

Each test gets its own temporary directory tree under a generated temporary directory in the system temp dir that is removed at the end of the run, or under the path given to `--temp-dir` on the command line. If a temporary directory is given explicitly, it will not be deleted at the end of the run.