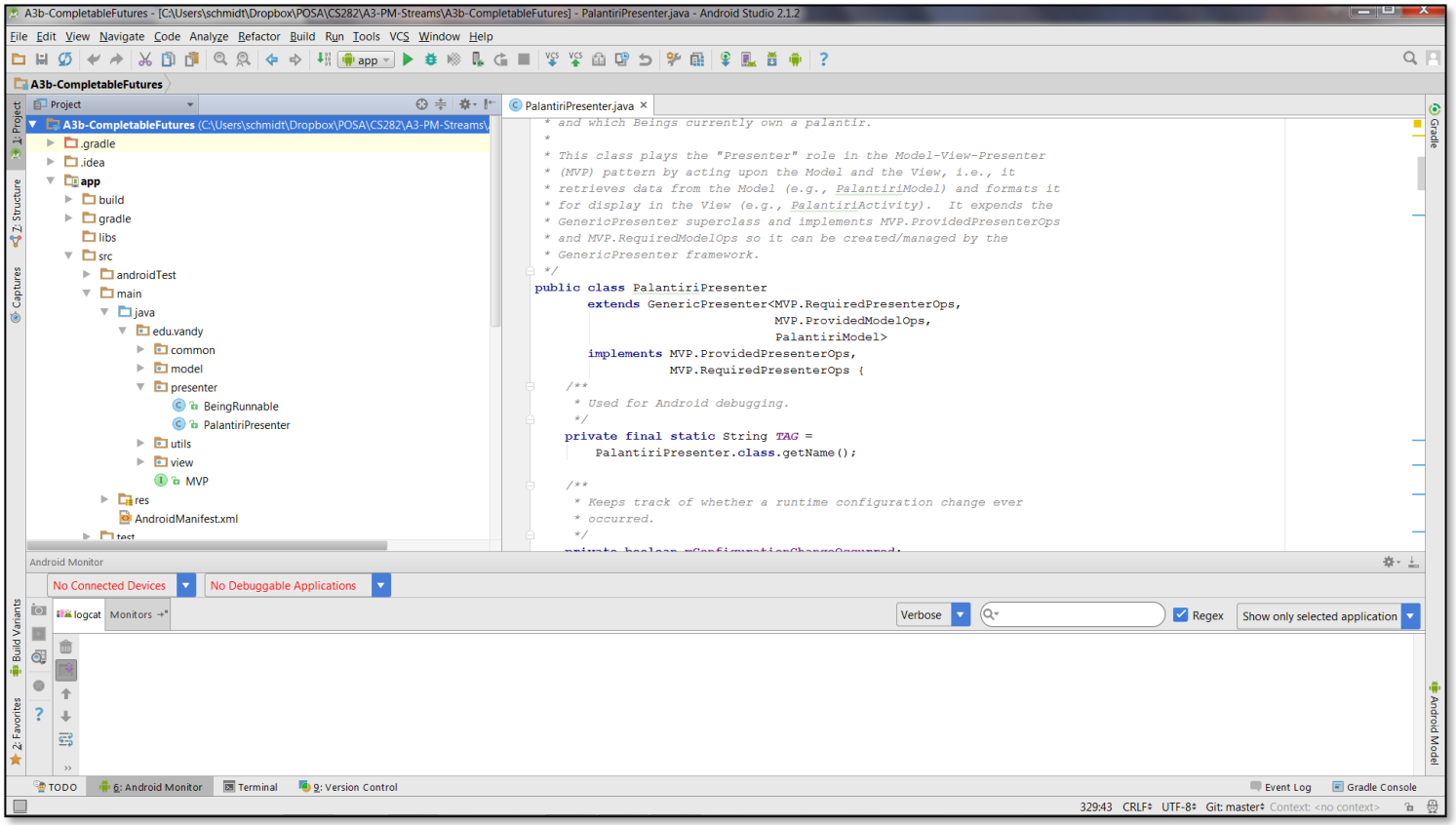
The Singleton Pattern

Implementation in C++

Douglas C. Schmidt

Learning Objectives in This Lesson

- Recognize how the *Singleton* pattern can be applied to centralize access to global resources.
- Understand the structure & functionality of the *Singleton* pattern.
- Know how to implement the *Singleton* pattern in C++.



Singleton example in C++

- Define a singleton class to handle command-line option processing.

```
class Options {
public:
  static Options *instance();


  // Parse command-line arguments & sets values as follows.
  bool parse_args(int argc, char *argv[]);
  bool verbose() const; // True if running in verbose mode.
  ...


private:
  Options();

  static Options *instance_;

  bool verbose_;
  ...
```

Singleton example in C++

- Define a singleton class to handle command-line option processing.

```
class Options {
public:
  static Options *instance();


  // Parse command-line arguments & sets values as follows.
  bool parse_args(int argc, char *argv[]);
  bool verbose() const; // True if running in verbose mode.
  ...


private:
  Options();

  static Options *instance_;

  bool verbose_;
  ...
```

Holds the one & only instance

Singleton example in C++

- Define a singleton class to handle command-line option processing.

```
class Options {
public:
  static Options *instance();

  // Parse command-line arguments & sets values as follows.
  bool parse_args(int argc, char *argv[]);
  bool verbose() const; // True if running in verbose mode.
  ...

private:
  Options();

  static Options *instance_;

  bool verbose_;
  ...
```

```
if (instance_ == nullptr)
  instance_ = new Options;
return instance_;
```

See en.wikipedia.org/wiki/Lazy_initialization

Singleton example in C++

- Define a singleton class to handle command-line option processing.

```
class Options {
public:
  static Options *instance();


  // Parse command-line arguments & sets values as follows.
  bool parse_args(int argc, char *argv[]);
  bool verbose() const; // True if running in verbose mode.
  ...
```

These non-static methods are called on the singleton instance

```
private:
  Options();

  static Options *instance_;

  bool verbose_;
  ...
```

Singleton example in C++

- Define a singleton class to handle command-line option processing.

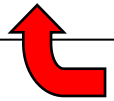
```
class Options {
public:
  static Options *instance();


  // Parse command-line arguments & sets values as follows.
  bool parse_args(int argc, char *argv[]);
  bool verbose() const; // True if running in verbose mode.
  ...


private:
  Options();

  static Options *instance_;

  bool verbose_;
  ...
```



Non-static fields

Singleton example in C++

- Define a singleton class to handle command-line option processing.

```
class Options {
public:
  static Options *instance();


  // Parse command-line arguments & sets values as follows.
  bool parse_args(int argc, char *argv[]);
  bool verbose() const; // True if running in verbose mode.
  ...


private:
  Options();

  static Options *instance_;

  bool verbose_;
  ...
```

Private constructor prevents multiple instances of an object from being created