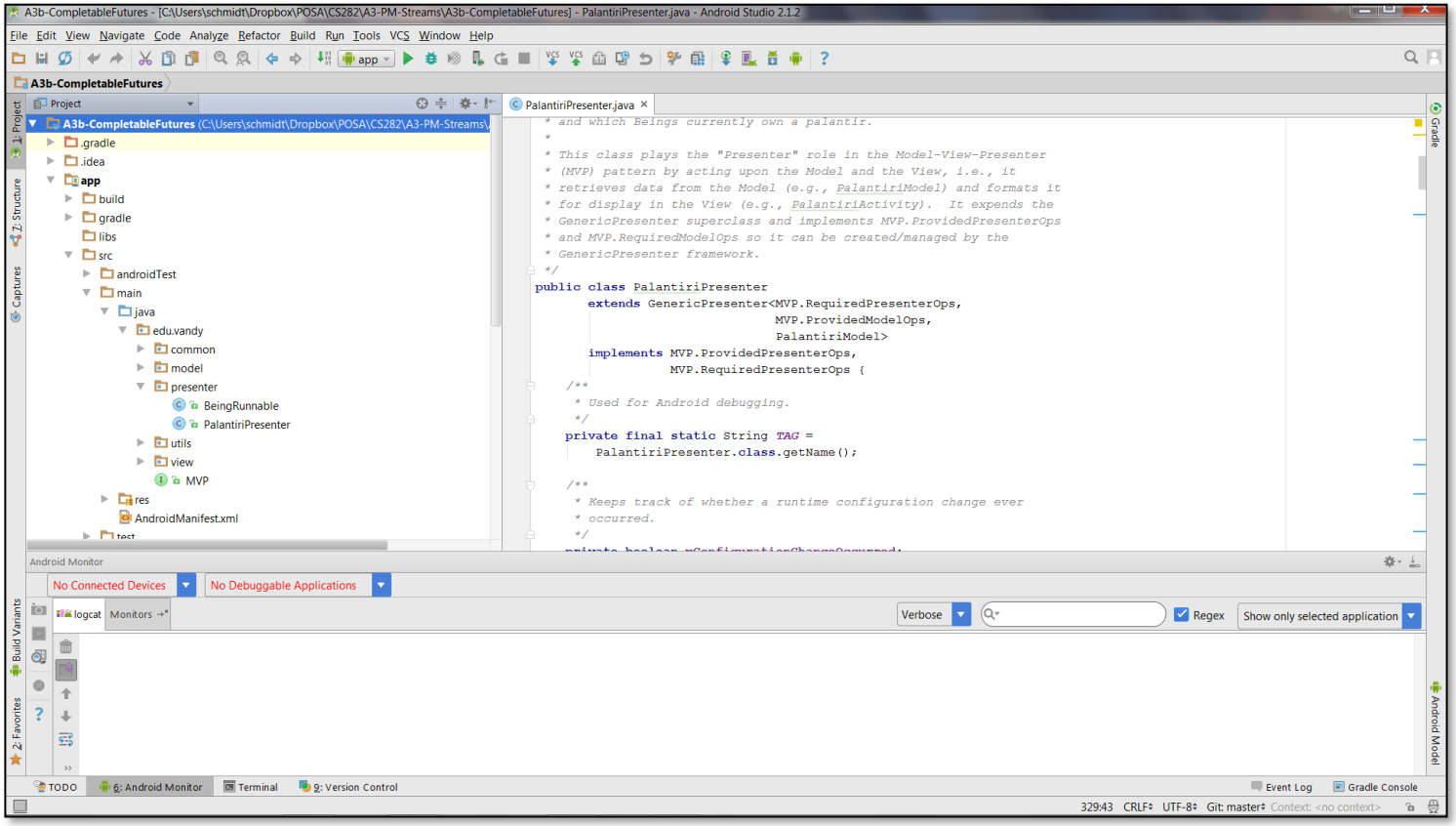
The Factory Method Pattern

Implementation in C++

Douglas C. Schmidt

Learning Objectives in This Lesson

- Recognize how the *Factory Method* pattern can be applied to extensibly create variabilities in the expression tree processing app.
- Understand the structure & functionality of the *Factory Method* pattern.
- Know how to implement the *Factory Method* pattern in C++.



Douglas C. Schmidt

Implementing the Factory Method Pattern in C++

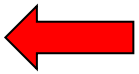
Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.

```
class User_Command_Factory_Impl {

  typedef User_Command (User_Command_Factory_Impl::*FACTORY_PTMF)
                          (const std::string &);
  typedef std::map<std::string, FACTORY_PTMF> COMMAND_MAP;

  COMMAND_MAP command_map_;

  User_Command_Factory_Impl(Tree_Context &tree_context)
    : command_map_, tree_context(tree_context) {
    command_map_["format"] = &User_Command_Factory_Impl
                               ::make_format_command;
    command_map_["expr"] = &User_Command_Factory_Impl
                             ::make_expr_command;
     ...
  }
```

Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.



We apply *Command* to initialize `User_Command_Factory_Impl`.

```
class User_Command_Factory_Impl {
  typedef User_Command (User_Command_Factory_Impl::*FACTORY_PTMF)
                              (const std::string &);
  typedef std::map<std::string, FACTORY_PTMF> COMMAND_MAP;

  COMMAND_MAP command_map_;

  User_Command_Factory_Impl(Tree_Context &tree_context)
    : command_map_, tree_context(tree_context) {
    command_map_["format"] = &User_Command_Factory_Impl
                                ::make_format_command;
    command_map_["expr"] = &User_Command_Factory_Impl
                               ::make_expr_command;
     ...
  }
```

Factory Method example in C++

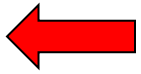
- `User_Command_Factory_Impl` creates `User_Commands`.


```
class User_Command_Factory_Impl {

  typedef User_Command (User_Command_Factory_Impl::*FACTORY_PTMF)
                              (const std::string &);
  typedef std::map<std::string, FACTORY_PTMF> COMMAND_MAP;

  COMMAND_MAP command_map_;

  User_Command_Factory_Impl(Tree_Context &tree_context)
    : command_map_, tree_context(tree_context) {
    command_map_["format"] = &User_Command_Factory_Impl
                                ::make_format_command;
    command_map_["expr"] = &User_Command_Factory_Impl
                               ::make_expr_command;
     ...
  }
```

Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.

```
class User_Command_Factory_Impl {

  typedef User_Command (User_Command_Factory_Impl::*FACTORY_PTMF)
                              (const std::string &);
  typedef std::map<std::string, FACTORY_PTMF> COMMAND_MAP;

  COMMAND_MAP command_map_;

  User_Command_Factory_Impl(Tree_Context &tree_context)
    : command_map_, tree_context(tree_context) {
    command_map_["format"] = &User_Command_Factory_Impl
                                ::make_format_command;
    command_map_["expr"] = &User_Command_Factory_Impl
                               ::make_expr_command;
     ...
  }
```



Map strings to factory commands that create `User_Command` objects

Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.

```
class User_Command_Factory_Impl {

  typedef User_Command (User_Command_Factory_Impl::*FACTORY_PTMF)
                              (const std::string &);
  typedef std::map<std::string, FACTORY_PTMF> COMMAND_MAP;

  COMMAND_MAP command_map_;

  User_Command_Factory_Impl(Tree_Context &tree_context)
    : command_map_, tree_context(tree_context) {
    command_map_["format"] = &User_Command_Factory_Impl
                                ::make_format_command;
    command_map_["expr"] = &User_Command_Factory_Impl
                               ::make_expr_command;
    ...
  }
```

Pointers-to-member-functions used to create `User_Command` objects

We apply the *Command* pattern to define a factory method that creates a command!

Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.

```
class User_Command_Factory_Impl {
```

The factory method

```
  User_Command make_command (const string &input) {
    string::size_type space_pos = input.find (' ');
    string parameters = input.substr (space_pos + 1);
    string command_keyword = input.substr (0, space_pos);



    auto iter = command_map_.find (command_keyword);

    if (iter == command_map_.end ())
      return User_Command_Factory_Impl
                          ::make_quit_command (parameters);
    else
      return (this->*iter->second) (parameters);
   }
```

The factory method uses a map to find/execute the command that makes a command.

Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.

```
class User_Command_Factory_Impl {


  User_Command make_command (const string &input) {
    string::size_type space_pos = input.find (' ');
    string parameters = input.substr (space_pos + 1);
    string command_keyword = input.substr (0, space_pos);
```

Parse inputstring to get command & params 

```
    auto iter = command_map_.find (command_keyword);

    if (iter == command_map_.end ())
      return User_Command_Factory_Impl
                          ::make_quit_command (parameters);
    else
      return (this->*iter->second) (parameters);
   }
```

Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.

```
class User_Command_Factory_Impl {


  User_Command make_command (const string &input) {
    string::size_type space_pos = input.find (' ');
    string parameters = input.substr (space_pos + 1);
    string command_keyword = input.substr (0, space_pos);
```

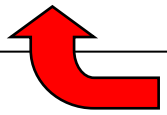
Try to find a pre-allocated factory command

```
    auto iter = command_map_.find (command_keyword);

    if (iter == command_map_.end ())
      return User_Command_Factory_Impl
                          ::make_quit_command (parameters);
    else
      return (this->*iter->second) (parameters);
  }
```

Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.

```
class User_Command_Factory_Impl {


  User_Command make_command (const string &input) {
    string::size_type space_pos = input.find (' ');
    string parameters = input.substr (space_pos + 1);
    string command_keyword = input.substr (0, space_pos);



    auto iter = command_map_.find (command_keyword);

    if (iter == command_map_.end ())
      return User_Command_Factory_Impl
                          ::make_quit_command (parameters);
    else
      return (this->*iter->second) (parameters);
   }
```



If found, execute it to make a command

Factory Method example in C++

- `User_Command_Factory_Impl` creates `User_Commands`.

```
class User_Command_Factory_Impl {


  User_Command make_command (const string &input) {
    string::size_type space_pos = input.find (' ');
    string parameters = input.substr (space_pos + 1);
    string command_keyword = input.substr (0, space_pos);



    auto iter = command_map_.find (command_keyword);

    if (iter == command_map_.end ())
      return User_Command_Factory_Impl
                          ::make_quit_command (parameters);
    else
      return (this->*iter->second) (parameters);
   }
```

Otherwise, user gave un-supported request, so quit