Date:.....

## ITEM 17: Store newed objects in smart pointers in standalone statements.

Eg →
```
int priority();
void processWidget(std::shared_ptr<Widget> pw,
                   int priority);
```

Call to process widget

```
processWidget(new Widget, priority());
```
→ won't compile

since new Widget do not return a type of std::shared_ptr.

```
processWidget(std::shared_ptr<Widget>
    (new Widget), priority());
```
→ will Compile.

Before process Widget can be called, compilers must generate code to do these:
1. Call priority
2. Execute "new widget"
3. Call std::shared constructor.

(In order Right to left)

Date:

But compiler can choose to perform the steps in any way (to generate more efficient code), we may end up with below sequence of operatn.

1. Execute "new Widget"
2. Call priority
3. Call tr1::shared constructor.

But consider if priority() yields an exceptn.
new Widget is already created thus a memory leak scenario can happen.

To avoid this, use seperate statement to create object and store it in a smart pointer, the pass it to functn

```
tr1::shared_ptr<Widget> pw(new Widget);
processWidget(pw, priority());
// no memory leak.
```