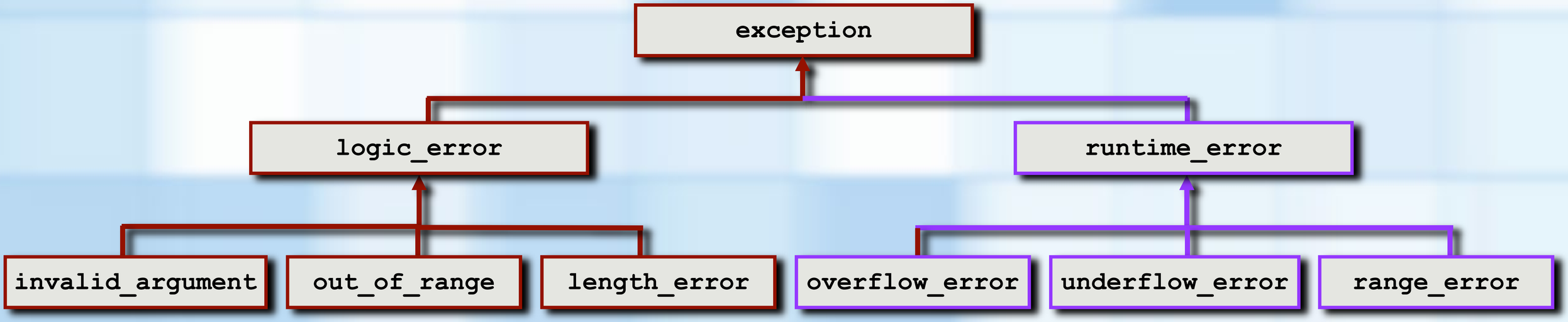
C++ Exceptions

Exception Basics

- **An unusual event or circumstance that interrupts execution of a program**
  - An object called an **exception** is created when this event or circumstance occurs.
  - Programmers can create their own exception objects.
  - A method that creates an exception object **throws** the exception.
    - C++ can through any type
  - Calling method or function can **handle** an exception
    - detect and react to it

Exception Basics

- C++ Exceptions Classes



Pearson

Throwing an Exception

- **Throwing an Exception**
  - Test for an exceptional or error condition.
  - **throw** an exception if the condition exists.
  - if a function cannot throw an exception, use the **noexcept** operator in the function header:

```
int myFunction( ) noexcept
{
  // function implementation
}
```

```
// Function to search an array of PlainBox<string> objects
// for a specific target item
// Returns PlainBox containing the item.
PlainBox<std::string> findBox(PlainBox<std::string> boxes[],
                              int size, std::string target)
{
   int index = 0;
   bool found = false;
   while (!found && (index < size))
   {
      if (target == boxes[index].getItem())
         found = true;
      else
         index++;
   } // end while

   if (!
         throw std::logic_error("Target not found");

   return boxes[index];
} // end findBox
```

Pearson

CATCHING AN EXCEPTION

```
// Create and initialize an array of boxes
PlainBox<std::string> myBoxes[5];
myBoxes[0] = PlainBox<std::string>("ring");
myBoxes[1] = PlainBox<std::string>("hat");
myBoxes[2] = PlainBox<std::string>("shirt");
myBoxes[3] = PlainBox<std::string>("sock");
myBoxes[4] = PlainBox<std::string>("shoe");
PlainBox<std::string> foundBox;

try
{
   foundBox = findBox(myBoxes, 5, "glasses");
}

std::cout << foundBox.getItem();
```

Using an Exception

```
// Create and initialize an array of boxes
PlainBox<std::string> myBoxes[5];
myBoxes[0] = PlainBox<std::string>("ring");
myBoxes[1] = PlainBox<std::string>("hat");
myBoxes[2] = PlainBox<std::string>("shirt");
myBoxes[3] = PlainBox<std::string>("sock");
myBoxes[4] = PlainBox<std::string>("shoe");
PlainBox<std::string> foundBox;

// Try to find a box containing glasses
try
{
   foundBox = findBox(myBoxes, 5, "glasses");
}
catch(std::logic_error logErr)
{
   std::cout << logErr.what() << std::endl;
   foundBox = PlainBox<std::string>("nothing");
}

std::cout << foundBox.getItem();
```

```
// Function to search an array of PlainBox<string> objects
// for a specific target item
// Returns PlainBox containing the item.
PlainBox<std::string> findBox(PlainBox<std::string> boxes[],
                              int size, std::string target)
{
   int index = 0;
   bool found = false;
   while (!found && (index < size))
   {
      if (target == boxes[index].getItem())
         found = true;
      else
         index++;
   } // end while

   if (!found)
         throw std::logic_error("Target not found in a box!");

   return boxes[index];
} // end findBox
```

Pearson

Using an Exception

```
// Create and initialize an array of boxes
PlainBox<std::string> myBoxes[5];
myBoxes[0] = PlainBox<std::string>("ring");
myBoxes[1] = PlainBox<std::string>("hat");
myBoxes[2] = PlainBox<std::string>("shirt");
myBoxes[3] = PlainBox<std::string>("sock");
myBoxes[4] = PlainBox<std::string>("shoe");
PlainBox<std::string> foundBox;

// Try to find a box containing glasses
try
{
   foundBox = findBox(myBoxes, 5, "shoe");
}
catch(std::logic_error logErr)
{
   std::cout << logErr.what() << std::endl;
   foundBox = PlainBox<std::string>("nothing");
}

std::cout << foundBox.getItem();
```

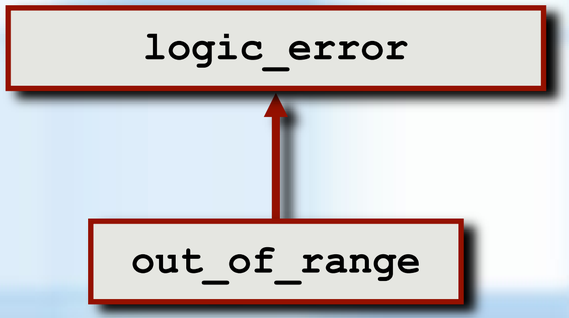
```
// Function to search an array of PlainBox<string> objects
// for a specific target item
// Returns PlainBox containing the item.
PlainBox<std::string> findBox(PlainBox<std::string> boxes[],
                              int size, std::string target)
{
   int index = 0;
   bool found = false;
   while (!found && (index < size))
   {
      if (target == boxes[index].getItem())
         found = true;
      else
         index++;
   } // end while

   if (!found)
         throw std::logic_error("Target not found in a box!");

   return boxes[index];
} // end findBox
```

Handling Multiple Exceptions

- Place `catch`-blocks in reverse order of class hierarchy.




```
std::string str = "Sarah";
try
{
   str.substr(99, 1);
}
catch (std::out_of_range e)
{
  std::cout << "out_of_range exception caught" << std::endl;
}
catch (std::logic_error e)
{
  std::cout << "Something else was caught" << std::endl;
}
```