Templates

- Overloaded functions perform similar operations that involve different program logic on different data types.

- If the program logic and operations are identical for each data type, this may be done by templates.
The programmer defines a single class template.

Given the argument types provided in calls to the constructor, compiler generates separate classes to handle each type of call appropriately.

A class template defines a whole family of solutions.

A template begins with template followed by list of types.

Each type is preceded by the keyword class.
Exceptions

- Errors can be dealt with at the places in the code where the error has occurred.

- Exception handling enables the programmer to remove error-handling code from the “main line”.

- Exception handling is used in situations in which the system can recover from the error causing the exception, or an orderly cleanup is required.

- The recovery procedure (exception handler) is typically used when the error will be dealt with by another part (i.e. a different scope) from that which detected the error.
Catching ALL Exceptions

- A `catch` followed by parenthesis enclosing an ellipsis means to catch all exceptions.

- For example:

```cpp
try {

} // end try  
catch ( ... )
{
  cout << "Catch ALL exceptions\n";
} // end catch
```
Standard Library Exception Hierarchy

- The hierarchy is headed by base class `exception` (defined in header file `<exception>`), which contains function `what()` that is overridden in each derived class to issue an appropriate message.

- Immediate derived classes of `exception` are `logic_error` and `runtime_error` (both defined in header `<stdexcept>`), each of which has several derived classes.

- `logic_error`: `domain_error`, `invalid_argument`, `length_error` and `out_of_range`.

- `runtime_error`: `range_error`, `overflow_error`, etc.

- Also derived from `exception` are `bad_alloc` thrown by `new`, and `bad_cast` thrown by `dynamic_cast`.

Shebuti Rayana (CS, Stony Brook University) (c) Pearson