PROGRAMMING WITH C++

LAB 9

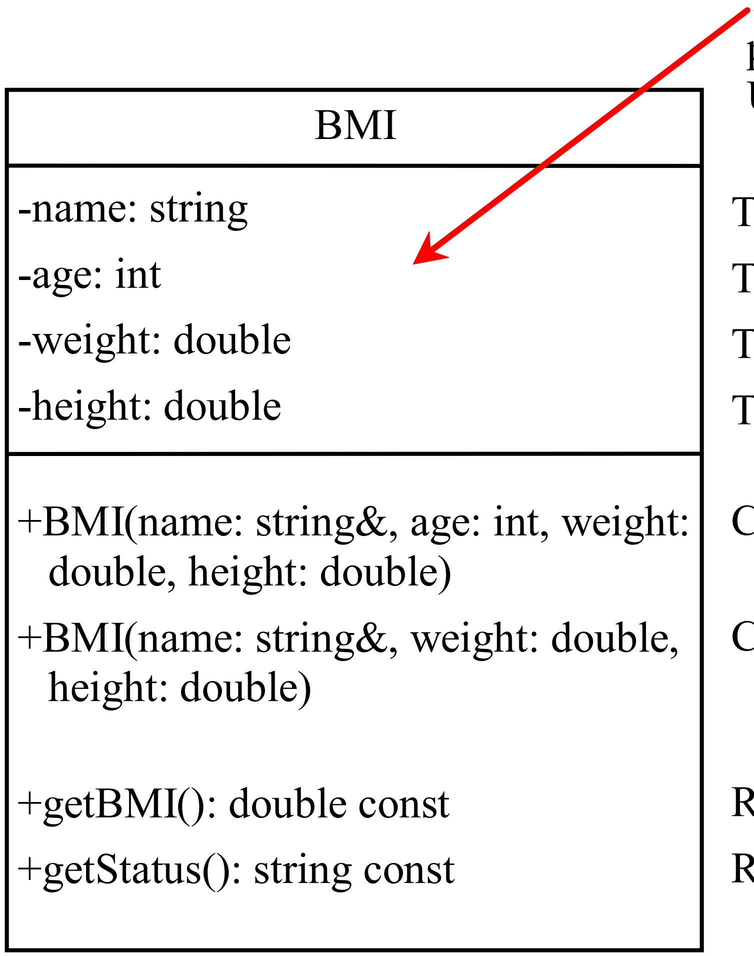
Xi'an Jiaotong-Liverpool University
西交利物浦大学

Example 1: The Loan Class

A specific loan can be viewed as an object of a Loan class. Interest rate, loan amount, and loan period are its data properties, and computing monthly payment and total payment are its functions. Let us use the Loan class as an example to demonstrate the creation and use of classes. Loan has the data fields annualInterestRate, numberOfYears, and loanAmount, and the functions getAnnualInterestRate, getNumberOfYears, getLoanAmount, setAnnualInterestRate, setNumberOfYears, setLoanAmount, getMonthlyPayment,and getTotalPayment, as shown in Figure below.

Loan	
-annualInterestRate: double	The annual interest rate of the loan (default: 2.5).
-numberOfYears: int	The number of years for the loan (default: 1)
-loanAmount: double	The loan amount (default: 1000).
+Loan()	Constructs a default Loan object.
+Loan(annualInterestRate: double, numberOfYears: int, loanAmount: double)	Constructs a loan with specified interest rate, years, and loan amount.
+getAnnualInterestRate(): double	Returns the annual interest rate of this loan.
+getNumberOfYears(): int	Returns the number of the years of this loan.
+getLoanAmount(): double	Returns the amount of this loan.
+setAnnualInterestRate(annualInterestRate: double): void	Sets a new annual interest rate to this loan.
+setNumberOfYears(numberOfYears: int): void	Sets a new number of years to this loan.
+setLoanAmount(loanAmount: double): void	Sets a new amount to this loan.
+getMonthlyPayment(): double	Returns the monthly payment of this loan.
+getTotalPayment(): double	Returns the total payment of this loan.

Example 2: The BMI Class



The get functions for these data fields are provided in the class, but omitted in the UML diagram for brevity.

BMI	
-name: string	The name of the person.
-age: int	The age of the person.
-weight: double	The weight of the person in pounds.
-height: double	The height of the person in inches.
+BMI(name: string&, age: int, weight: double, height: double)	Creates a BMI object with the specified name, age, weight, and height.
+BMI(name: string&, weight: double, height: double)	Creates a BMI object with the specified name, weight, height, and a default age 20.
+getBMI(): double const	Returns the BMI.
+getStatus(): string const	Returns the BMI status (e.g., Normal, Overweight, etc.)

Example 3: Array Of Objects

In Chapter 7, arrays of primitive type elements and strings were created. You can create arrays of any objects. For example, the following statement declares an array of 10 Circle objects:

Circle circleArray[10];

The name of the array is circleArray, and the no-arg constructor is called to initialize each element in the array. So, circleArray[0].getRadius() returns 1, because the no-arg constructor assigns 1 to radius. You can also use the array initializer to declare and initialize an array using a constructor with arguments. For example,

Circle circleArray[3] = {Circle(3), Circle(4), Circle(5)};

Write a program that demonstrates how to use an array of objects. The program summarizes the areas of an array of circles. It creates circleArray, an array composed of 10 Circle objects; it then sets circle radii with radius 1, 2, 3, 4, . . . , and 10 and displays the total area of the circles in the array.

Example 4: The Course Class

Suppose you need to process course information. Each course has a name and a number of students who take the course. You should be able to add/drop a student to/from the course. You can use a class to model the courses

Course	
-courseName: string	The name of the course.
-students: string*	An array of students who take the course. students is a pointer for the array.
-numberOfStudents: int	The number of students (default: 0).
-capacity: int	The maximum number of students allowed for the course.
+Course(courseName: string&, capacity: int)	Creates a Course with the specified name and maximum number of students allowed.
+~Course()	Destructor
+getCourseName(): string const	Returns the course name.
+addStudent(name: string&): void	Adds a new student to the course.
+dropStudent(name: string&): void	Drops a student from the course.
+getStudents(): string* const	Returns the array of students for the course.
+getNumberOfStudents(): int const	Returns the number of students for the course.

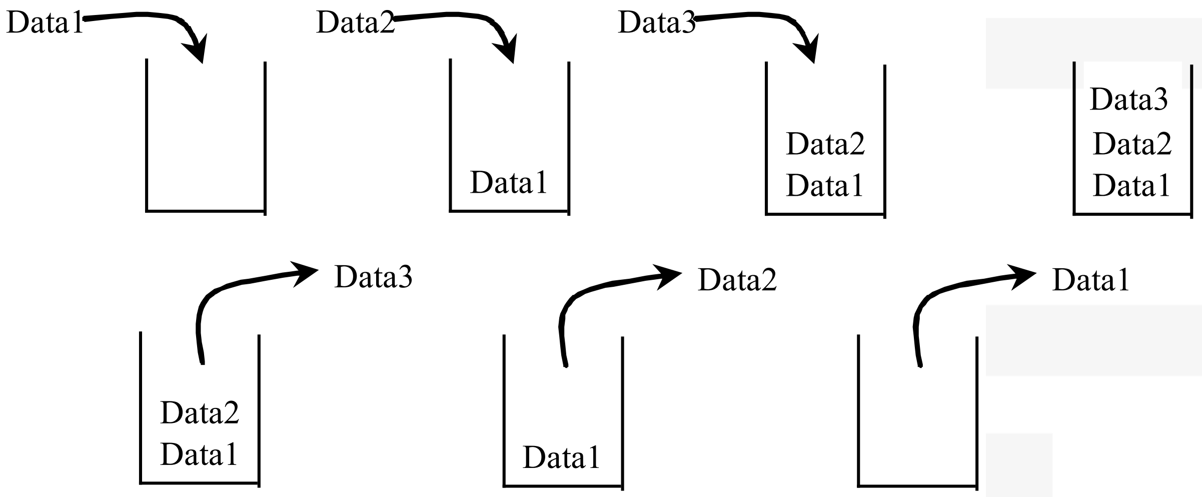
Example 5: The `StackOfIntegers` Class

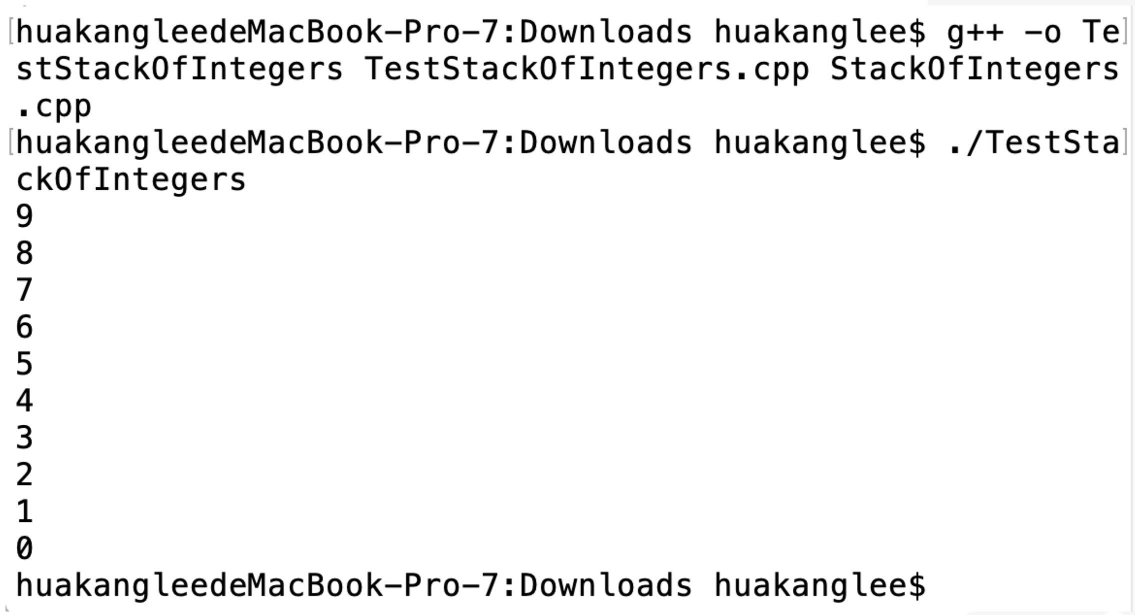
Recall that a stack is a data structure that holds data in a last-in, first-out fashion. You can define a class to model stacks. For simplicity, assume the stack holds the int values. So, name the stack class StackOfIntegers. The UML diagram for the class is shown bellow.

StackOfIntegers	
-elements[100]: int	An array to store integers in the stack.
-size: int	The number of integers in the stack.
+StackOfIntegers()	Constructs an empty stack.
+isEmpty(): bool const	Returns true if the stack is empty.
+peek(): int const	Returns the integer at the top of the stack without removing it from the stack.
+push(value: int): void	Stores an integer into the top of the stack.
+pop(): int	Removes the integer at the top of the stack and returns it.
+getSize(): int const	Returns the number of elements in the stack.

TestStackOfIntegers.cpp

```
#include <iostream>
#include "StackOfIntegers.h"
using namespace std;

int main()
{
  StackOfIntegers stack;

  for (int i = 0; i < 10; i++)
    stack.push(i);

  while (!stack.isEmpty())
    cout << stack.pop() << " ";

  return 0;
}
```





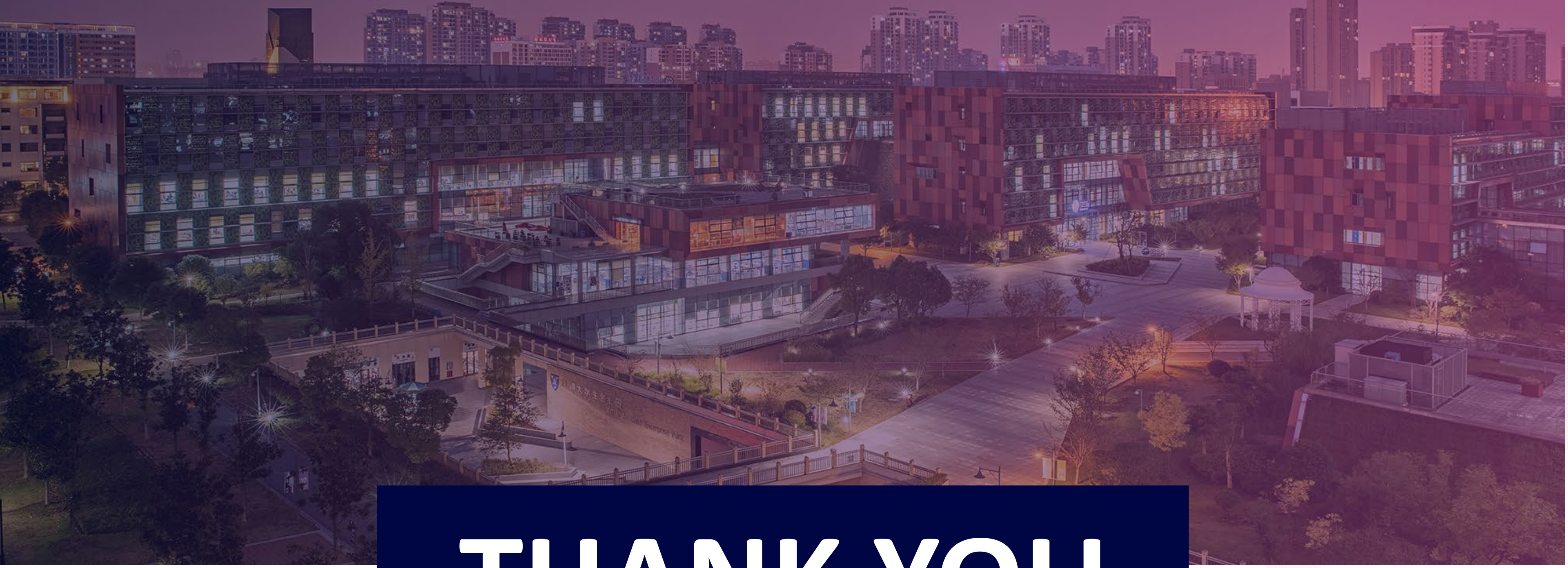
```
huakangleedeMacBook-Pro-7:Downloads huakanglee$ g++ -o Te
stStackOfIntegers TestStackOfIntegers.cpp StackOfIntegers
.cpp
huakangleedeMacBook-Pro-7:Downloads huakanglee$ ./TestSta
ckOfIntegers
9
8
7
6
5
4
3
2
1
0
huakangleedeMacBook-Pro-7:Downloads huakanglee$
```

StackOfIntegers.h

```
#ifndef STACK_H
#define STACK_H

class StackOfIntegers
{
public:
  StackOfIntegers();
  bool isEmpty() const;
  int peek() const;
  void push(int value);
  int pop();
  int getSize() const;

private:
  int elements[100];
  int size;
};

#endif
```

StackOfIntegers.cpp

```
#include "StackOfIntegers.h"
StackOfIntegers::StackOfIntegers()
{
  size = 0;
}
bool StackOfIntegers::isEmpty() const
{
  return size == 0;
}
int StackOfIntegers::peek() const
{
  return elements[size - 1];
}
void StackOfIntegers::push(int value)
{
  elements[size++] = value;
}
int StackOfIntegers::pop()
{
  return elements[--size];
}
int StackOfIntegers::getSize() const
{
  return size;
}
```



THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

Xi'an Jiaotong-Liverpool University
西交利物浦大學