Department of Statistics & Computer Science
University of Kelaniya
Academic Year – 2022/2023
COSC 12043 / BECS 12243 - Object Oriented Programming
Tutorial 11

---

1. Create a class with a method that prints "This is parent class" and its subclass with another method that prints "This is child class". Now, create an object for each of the classes and call,
   a. method of parent class by the object of the parent class
   b. method of child class by object of child class
   c. method of parent class by object of child class
   d. method of child class by the object of the parent class

   What is the output of each call above? Are there any errors that occurred? Give reasons for them.

2. Create a class named '**Member**' having the following members:

   Data members
   1 - Name
   2 - Age
   3 - Phone number
   4 - Address
   5 - Salary

   It also has a method named 'printSalary' which prints the salary of the members. Two classes, '**Employee**' and '**Manager**' inherit the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively.
   a. Assign name, age, phone number, address, salary, specialization, and department to an employee and a manager by making an object of both of these classes.
   b. Call print salary method for both objects.
   c. Print all the other details of both objects using reference variables.

3. You are designing an E-Learning Platform that supports two types of users: Students and Instructors, who share common attributes and behaviors while also having unique functionalities. To implement this, create a superclass named **User** with attributes such as name, and userID to represent the general properties of all users. The User class should include a method login(), which displays the message:

   `User <name> with ID <userID> has logged in.`

   Extend this superclass to create two subclasses: **Student** and **Instructor**. The Student subclass should include an additional attribute enrolledCourses, represented as an array of course names, and a method viewCourses() to display these courses in the format:

   `Enrolled Courses: course1, course2, ...`

   The Instructor subclass should include an attribute teachingCourses, also represented as an array of course names, and a method assignGrades(studentName, grade) that takes a student's name and a grade as inputs and prints:

   `Instructor <name> assigned grade <grade> to <studentName>.`

   Write a program to create and initialize objects for both Student and Instructor classes, call their respective methods, and verify if objects of one subclass can access methods specific to the other.

4. Design a Banking System that includes a superclass **Account** and two subclasses: **SavingsAccount** and **CurrentAccount**.
   a. Create an Account class with the following properties:
      accountNumber (String)
      accountHolderName (String)
      balance (double)
   b. Implement a constructor to initialize the accountNumber, accountHolderName, and balance.
   c. Create methods for:
      deposit(double amount): To deposit money into the account.
      displayAccountDetails(): To display the account number, account holder name, and balance.
   d. Create a SavingsAccount subclass that extends the Account class.
   e. Add an additional attribute:
      interestRate (double)

f. Create methods for:

withdraw(double amount): To withdraw money from the account, ensuring the balance is sufficient.

applyInterest(): To calculate and apply interest to the balance, use the formula:

Interest = balance * interestRate / 100

g. Create a CurrentAccount subclass that extends the Account class.

h. Add an attribute:

creditLimit (double)

i. Implement a method setCreditLimit(double limit) to set the credit limit.

j. Create a withdraw(double amount) method to allow withdrawals beyond the current balance, but only within the creditLimit.

k. In a main class (BankingSystem), create instances of SavingsAccount and CurrentAccount.

l. Test the following:

i. Create accounts for both SavingsAccount and CurrentAccount.

ii. Perform deposit and withdrawal operations.

iii. Apply interest on the SavingsAccount.

iv. Check if the CurrentAccount can withdraw beyond its balance as long as the credit limit is not exceeded.

v. Print the account details after each operation.

5. Design a **Calculator** class that performs basic arithmetic operations. The class should include overloaded methods for each operation (addition, subtraction, and division) to support both integer and double values.

6. Create a class **StringManipulator** with overloaded methods:

`String reverse(String s)` – Reverses a string.

`String reverse(String s, int n)` – Reverses the first n characters of a string.

`String reverse(String s, int start, int end)` – Reverses a substring from start to end.

Use the main method to test the overloaded methods.