University American College Skopje

Course: Object Programming

Virtual Functions. Polymorphism

Exercises

Prepared by: Adrijan Božinovski, Ph.D.

Assignment 1

- Create a base class **Base**, with the following fields:
  - Public:
    - virtual void f() function, which will display a certain message
- Create a class **Derived**, which publicly inherits from **Base**, and has the following additional fields:
  - Public:
    - a void f() function, which overrides the **Base** class' definition and will display a different message

Assignment 1

- In the main() function
  - Declare two Base pointers, x and y
  - Give one of the pointers (e.g. x) a new Base object's address
  - Through it, invoke the f() function
  - Give the other pointer (e.g. y) a new Derived object's address
  - Through the second pointer, invoke the f() function

Assignment 2

- Modify the main() function in Assignment 1, such that you will create two **Base** class objects, instead of pointers, plus an additional **Derived** class object
  - e.g. x and y should be objects instead of pointers and also a **Derived** class object, called z, should be created
- Assign the value of the **Derived** object to the second **Base** object and again invoke the f() function through both **Base** objects, as well as the **Derived** object
- What is the difference? Why is the result different?

Assignment 3

- Modify Assignment 1, such that the f() function in the **Base** class will no longer be virtual. Leave the pointers as they were

- What is the difference? Why is the result different?

Assignment 4

- Create a base class **Animal**, with the following fields:
  - Public:
    - string name
    - a parameterized constructor that takes one string parameter and sets the name field to the value of the parameter
    - a virtual string call() function, that returns the string “*none*”

Assignment 4

- Create a class **Lion**, which publicly inherits from **Animal**, and has the following additional fields:
  - Public:
    - a parameterized constructor, which takes one string parameter, and passes it along to the **Animal** class' constructor. It has no extra code
    - a string call() function, which returns the string "ROAR"

Assignment 4

- Create a class **Frog**, which publicly inherits from **Animal**, and has the following additional fields:
  - Public:
    - a parameterized constructor, which takes one string parameter, and passes it along to the **Animal** class' constructor. It has no extra code
    - a string call() function, which returns the string "CROAK"

Assignment 4

- Create a class **Dog**, which publicly inherits from **Animal**, and has the following additional fields:
  - Public:
    - a parameterized constructor, which takes one string parameter, and passes it along to the **Animal** class' constructor. It has no extra code
    - a string call() function, which returns the string "ARF ARF"

Assignment 4

- Create a class **SaraMountainDog**, which publicly inherits from **Dog**, and has the following additional fields:
  - Public:
    - a parameterized constructor, which takes one string parameter, and passes it along to the **Dog** class' constructor. It has no extra code
    - a string call() function, which returns the string "WOOF WOOF"
    - a void features() function, which outputs the string: "The Sara Mountain Dog ", then the name field, then " is the champion of Macedonia for 2010"

Assignment 4

- In the main() function
  - Declare a pointer to an **Animal** object
  - Give it the address of a new **Animal** object, with the name “Nessie”
  - Invoke the call() function through it
  - Give it the address of a new **Lion** object, with the name “Simba”
  - Invoke the call() function through it
  - Give it the address of a new **Frog** object, with the name “Kermit”
  - Invoke the call() function through it
  - Give it the address of a new **Dog** object, with the name “Rex”
  - Invoke the call() function through it
  - Give it the address of a new **SaraMountainDog** object, with the name “Sarko”
  - Invoke the call() function through it
  - Downcast it to a SaraMountainDog pointer, and invoke the features() function through this downcast version

Assignment 4

```
The sound of Nessie is *none*
The sound of Simba is ROAR
The sound of Kermit is CROAK
The sound of Rex is ARF ARF
The sound of Sarko is WOOF WOOF
The Sara Mountain Dog Sarko is the champion of Macedonia for 2010
Press any key to continue
```

Assignment 5

- Apply the following modifications to Assignment 4:
  1. Class **Lion** is derived from class **Animal** through protected inheritance
  2. Class **Frog** is derived from class **Animal** through private inheritance
  3. Class **Dog** is derived from class **Animal** through protected inheritance
  4. Class **SaraMountainDog** is derived from class **Dog** through private inheritance
- Apply additional modifications to the main() function, so that the program compiles correctly and gives the same output as in Assignment 4

Assignment 6

- Modify Assignment 5, so as to make class **Dog** derive from class **Animal** through private inheritance.

- What other modifications would be needed so that the program again compiles correctly and gives the same output?

Assignment 7

- Create a base class **Figure**, with the following fields:
  - Public:
    - a pure virtual double area() function;
- Create a class **Figure2D**, which inherits publicly from **Figure**, and has the following additional fields:
  - Public:
    - a pure virtual double circumference() function;
- Create a class **Figure3D**, which inherits publicly from **Figure**, and has the following additional fields:
  - Public:
    - a pure virtual double volume() function;

Assignment 7

- Create a class **Circle**, which inherits publicly from **Figure2D**, and has the following additional fields:
  - Private:
    - double r – it represents the radius of the circle;
  - Public:
    - a get() method for the radius
    - a parameterized constructor, that takes one double parameter and sets the value of the radius to the value of the parameter
    - a double circumference() function, which calculates the circumference of the given circle (a formula follows)
    - a double area() function, which calculates the area of the given circle (a formula follows)

Assignment 7

- Create a class **Sphere**, which inherits publicly from **Figure3D**, and has the following additional fields:
  - Private:
    - double r – it represents the radius of the sphere;
  - Public:
    - a get() method for the radius
    - a parameterized constructor, that takes one double parameter and sets the value of the radius to the value of the parameter
    - a double area() function, which calculates the area of the given circle (a formula follows)
    - a double volume() function, which calculates the volume of the given circle (a formula follows)

Assignment 7

- In the main() function:
  - Create a pointer to a **Figure** object, and a double radius variable
  - Input the radius variable through the keyboard
  - Create a new **Circle** object, using the **Figure** pointer
  - Using the pointer, display the radius, circumference and area of the circle. Use downcasting where necessary
  - *Perform memory clean-up after displaying the data*
  - Repeat the previous procedure to create a **Sphere** object, and display its radius, area and volume. Again, use downcasting where necessary, as well as memory clean-up

Assignment 7

	Circumference	Area	Volume
Circle	$2 \cdot r \cdot \pi$	$r^2 \cdot \pi$	N/A
Sphere	N/A	$4 \cdot r^2 \cdot \pi$	$4 \cdot r^3 \cdot \pi / 3$

```
Enter the radius of the circle: 1.1
A circle with the following features has been input:
Radius:          1.1
Circumference:   6.908
Area:            3.7994

Enter the radius of the sphere: 2.2
A sphere with the following features has been input:
Radius:          2.2
Area:            60.7904
Volume:          44.5796
Press any key to continue_
```