

**OVERALL DECISION: COMPETENT**

**Excellent. Well done!**

**Formative Assessment**

**Object-oriented Programming (115378)**

*Hello and welcome to the assessment. Here you’ll prove to the world just how much you know and understand about what you’ve just learnt in the learner guides. This is an important part of your time at Umuzi because once this is done, you’ll be certified! So please, take this time to learn everything you can! Take a look at some pointers below with regard to answering the questions…*

* *Be specific*
* *Write professionally - no shorthand!*
* *Your answers must be original and come from your brain and your brain only.*
* *No copy/paste tricks! Our markers have seen it all and will know if you’re taking shortcuts.*
* *Remember, sloppy or poor work will be sent back to you to do again, so do it properly the first time and you’ll be done in no time.*
* *Ask for help at any time. Ask your friends, a manager, anybody!!*
* *Don’t skip any questions! You must do them all!*
* *You’ll see two boxes after each question - one for your answer and one for the marker’s comments. DO NOT delete the marker’s comments if you are required to resubmit your work after the first attempt. Should you have to do it again you will see a new box* ***under*** *the marker’s comments, so fill that one out in* ***BLUE****. Remember!! It’s not the end of the world if you have to resubmit. You’re here to learn, so don’t beat yourself up if you don’t get it right on the first go. Obviously, try your best to get it right on the first attempt, but if not, you have another chance to do it properly!*

*Ok, and that’s that! Time to get to it! Good luck, have fun and enjoy! :)*

**Enter your name and surname below**

|  |
| --- |
| **Sinethemba Zulu** |

**1.** **Describe inheritance in Object-Oriented Programming and provide an example of when it should be used [5 Marks](5 SO:1 AC:1-2)**

**Your answer below**

|  |
| --- |
| In object-oriented programming, inheritance allows new objects to take over the properties of existing objects. A class that is used as the basis of inheritance is called a superclass or a foundation class. A class that is inherited from a superclass is called a subset or a derived class. E.g****    # A Python program to demonstrate inheritance ****    # Base or Super class. Note object in bracket.  # (Generally, object is made ancestor of all classes)  # In Python 3.x "class Person" is  # equivalent to "class Person(object)"  class Person(object): ****    # Constructor  def \_\_init\_\_(self, name):  self.name = name ****    # To get name  def getName(self):  return self.name ****    # To check if this person is employee  def isEmployee(self):  return False****      # Inherited or Sub class (Note Person in bracket)  class Employee(Person):    # Here we return true  def isEmployee(self):  return True****    # Driver code  emp = Person("Geek1") # An Object of Person  print(emp.getName(), emp.isEmployee())****    emp = Employee("Geek2") # An Object of Employee  print(emp.getName(), emp.isEmployee())**** |

**Marker’s Comments**

|  |
| --- |
| **Correct** |

**2. Describe polymorphism in Object-Oriented Programming and provide an example of when it should be used [5 Marks](5 SO:2 AC:1-2)**

**Your answer below**

|  |
| --- |
| Polymorphism simply means 'one function, many forms.' Polymorphism is achieved in both compile time and run time. Compile time polymorphism is achieved by overloading, while run time polymorphism is accomplished by overloading. ****  E.g  class Cat:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age****  def info(self):  print(f"I am a cat. My name is {self.name}. I am {self.age} years old.") ****  def make\_sound(self):  print("Meow")  ****  class Dog:  def \_\_init\_\_(self, name, age):  self.name = name  self.age = age  ****  def info(self):  print(f"I am a dog. My name is {self.name}. I am {self.age} years old.")  def make\_sound(self):  print("Bark")  **** |

**Marker’s Comments**

|  |
| --- |
| **Correct** |

**3. What is a class? Include in your description how it incorporates abstraction and encapsulation in its design. [6 Marks](6 SO:3 AC:1-4)**

**Your answer below**

|  |
| --- |
| A class is a code template for creating objects. Objects have member variables and have behaviour associated with them. In python a class is created by the keyword class. ****  Encapsulation is used to hide code and data in a single unit to shield it from outside the world. Class is the best form of encapsulation. Abstraction, on the other hand, involves just displaying the required information to the intended consumer. **** |

**Marker’s Comments**

|  |
| --- |
| **Correct** |

**4. What two features make something an “abstract class”? (4 SO:4 AC:1-3)**

**Your answer below**

|  |
| --- |
| * They cannot be instantiated (made into objects), **** * They can have special methods called “abstract methods”. **** |

**Marker’s Comments**

|  |
| --- |
| **Correct** |

**5. What is a design pattern? Describe it in general and give a specific example [6 Marks](6 SO:5 AC:1-3)**

**Your answer below**

|  |
| --- |
| **Design pattern** is a general repeatable approach to a common problem in program design. A design pattern is not a completed design that can be specifically translated into a code. It is a summary or template of how to approach a problem that can be found in several different scenarios. ****  For example, a single-tone design pattern requires the use of a single object, such that all developers familiar with a single design pattern use a single object, so they may inform each other that a single-tone pattern accompanies a program. **** |

**Marker’s Comments**

|  |
| --- |
| **Correct** |