Centro de Investigación y de Estudios Avanzados del IPN
Unidad Tamaulipas

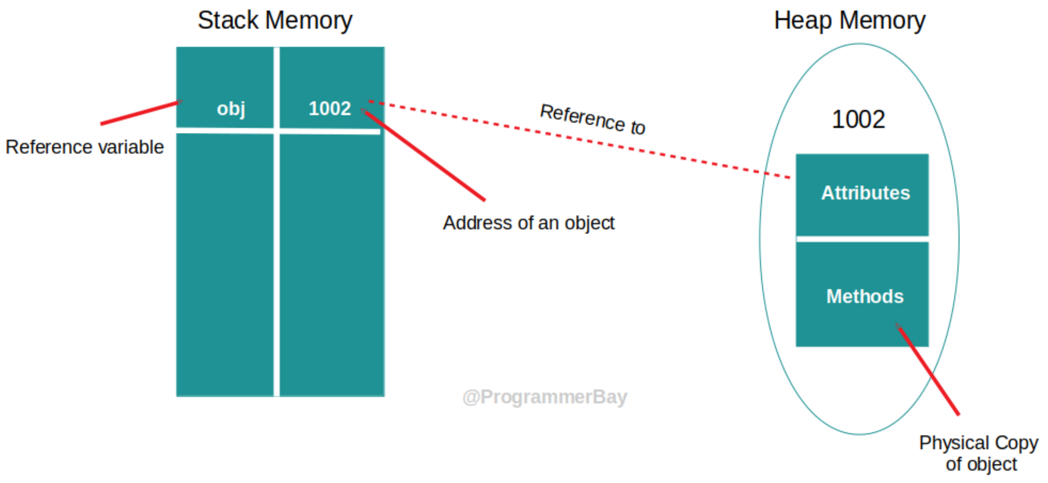
Assignment #5

Student: *Luis Alberto Ballado Aradias*
Course: *Tecnologías Computacionales (Sep - Dec 2022)*
Professor: *Dr. Edwin Aldana Bobadilla*
November 15, 2022

Memory Allocation

The underlying OS provides some memory to every program that runs and the memory can be divided into two areas.

- STACK - used to store temporary variables, primitive data types etc. A block in the stack exists for a variable only as long as the variable exists. After that, the block data is erased and it can be used for storing another variable.
- HEAP - is a special Tree-based data structure in which the tree is a complete binary tree. In heap memory, a physical copy of an object is stored and the reference or address of this copy in the variable of the associated class. Objects that can no longer be referenced are cleaned up by Garbage collector in JAVA language.



In every program there are four entities that need to live on this memory

- Local Variables - Are the variables that are declared inside a method and they live on the stack, they are temporary and they live on the stack as long as the method is executing, so any current executing method along with its local variables lives on the stack until it is done.
- Methods - is a procedure associated with a message and an object.

- Objects - Live on the heap and since we know that the instance variables belong to their respective objects instance variables live inside the object on a heap
- Instance Variables - is a variable which is declared in a class but outside of constructructors, methods, or blocks. Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.

For example:

A Car Object (**Car myCar = new Car();**) which has certain instance variables like: gears, wheels, height. So in heap memory the object always has a sufficient amount of memory to contain all its instance variables.

Object creation is a two step process **Declaration** (**Car myCar**) and **Instantiation** (**new Car();**)
In my declaration I have declared a reference variable of the type Car. I have used two keywords here the reference variable and the type car. My Car is called a reference variable, because it does not hold the actual object, just a reference or the address of the memory pointing to the actual object, and car is called a type because car is a class.

A class is a logical framework that defines the relationship between it is members, so class is nothing but a user-defined data type and of course I can use this data type to create my reference variable instantiation. The new keyword actually creates the object and the memory is allocated to this object for which the reference is assigned to my Car reference variable, there is one thing that is important here. Since the new keyword allocates memory only at runtime, so the memory gets allocated when the program is executing.

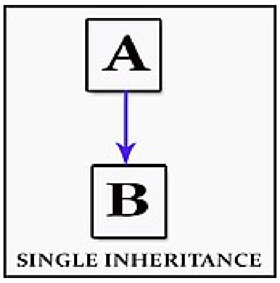
Constructors

- Every class has a constructor.
- A compiler provides the default constructor if no constructor is present in the code.
- Constructor is called during the lifecycle of an object.
- Constructor provides a fully initialised usable object immediately upon object creation.

There are various types of inheritance, based on paradigm and specific language.

Single inheritance

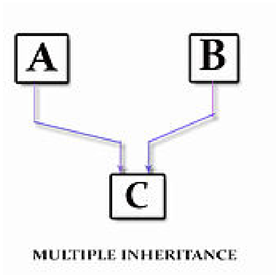
where subclasses inherit the features of one superclass. A class acquires the properties of another class.



Single inheritance enables a derived class to inherit properties and behavior from a single parent class. It allows a derived class to inherit the properties and behavior of a base class, thus enabling code reusability as well as adding new features to the existing code. This makes the code much more elegant and less repetitive. Inheritance is one of the key features of object-oriented programming (OOP). **Single inheritance** is safer than multiple inheritance if it is approached in the right way. It also enables a derived class to call the parent class implementation for a specific method if this method is overridden in the derived class or the parent class constructor.

Multiple inheritance

where one class can have more than one superclass and inherit features from all parent classes.



Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. The constructors of inherited classes are called in the same order in which they are inherited. For example, in the following program, B's constructor is called before A's constructor. A class can be derived from more than one base class. (i.e. A CHILD class is derived from FATHER and MOTHER class.)

Unlike some other popular object oriented programming languages like C++, java doesn't provide support for multiple inheritance in classes. Java doesn't support multiple inheritances in classes because it can lead to diamond problem and rather than providing some complex way to solve it, there are better ways through which we can achieve the same result as multiple inheritances.