Program organization inside the computer memory

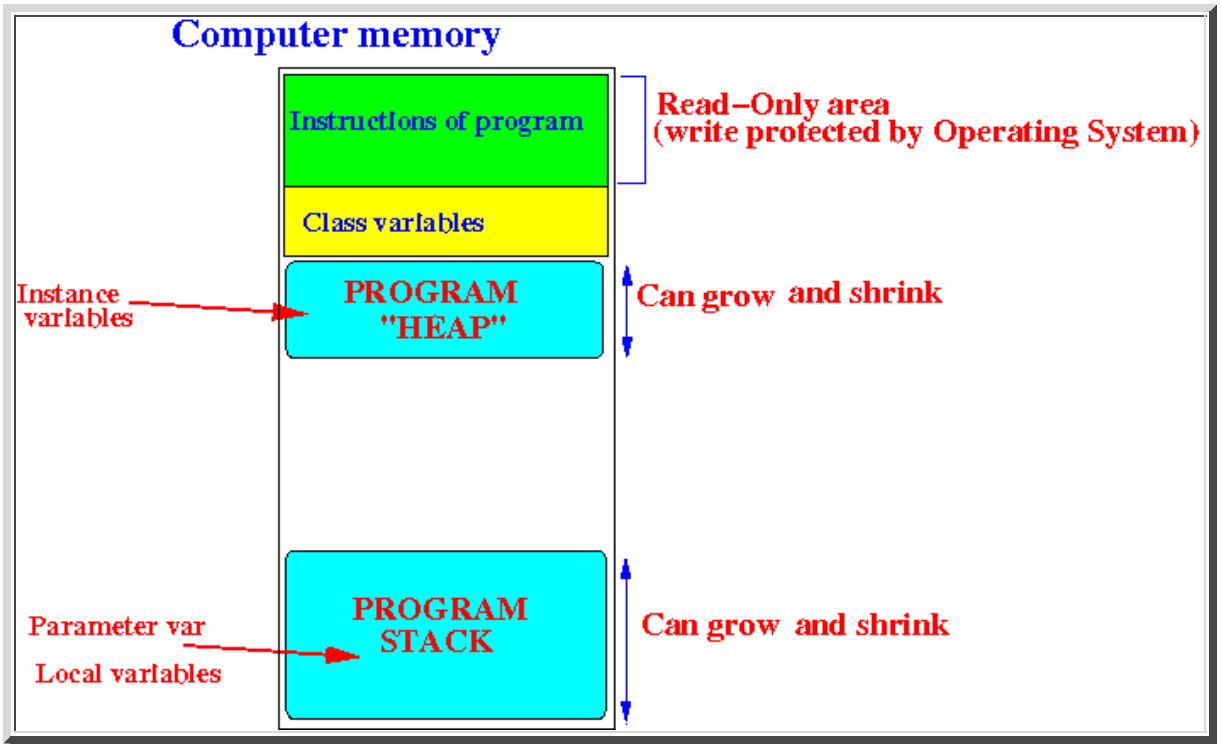
- **Computer programs and computer main memory**
  - **Computer Program:**
    - A **computer program** consists of:
      1. **Program *instructions*** (**statements** in the **computer program**)
      2. **Program *variables***
  - **Storing** a **computer program** in **memory**:
    - **Each program instruction** in a **computer program** is **stored** in a **specific location (*address*)** in **main memory**
    - **Each variables** defined in a **computer program** is **stored** in a **specific location (address)** in **main memory**
- **Kinds of variables**
  - **Recall** that **Java** has **4 different *kinds*** (**not *type***) of **variables**:
    - **Class variables**
    - **Instance variables**
    - **Local variables**
    - **Parameter variables**
  - **Recall** on **how** to **recognize** the different **kinds** of **variables**:

```
public class ExampleClass
{
    static int  x;          // Keyword static defines a Class variable

    int y;                  // Absent of static defines an Instance var

    public int Method1( int a )  // a is a parameter variable
    {
       int b;                    // Local variables are defined inside a method

       ....
    }
 }
```

- **Life time of variables**
  - **Recall** from **CS170** --- **Life time** of **variables**:
    - **Life time** of a **variable** = the **(time) period** between:
      - The **moment** that the **variable begins** to **exists**
      - The **moment** that the **variable** ***ceases*** to **exists**
- **The life time of the different kinds of variables**
  - The **life time** of **Class variables**:
    - **Begins:** from the ***execution*** of the **Java program**
    - **Ends:** at the ***termination*** of the **Java program**
  - The **life time** of **Instance variables**:
    - **Begins:** when the **instance variable** is ***created*** by a **`new` operation**
    - **Ends:** when the **instance variable** becomes ***garbage*** (i.e., **inaccessible**)
  - The **life time** of **parameter variables**:
    - **Begins:** at the ***start*** of the **method execution**
    - **Ends:** at the ***termination*** **(= exit)** of the **method execution**
  - The **life time** of **local variables**:
    - **Begins:** at the ***start*** of the **method execution**
    - **Ends:** at the ***termination*** **(= exit)** of the **method execution**
- **Storing the different kinds of variables in memory**
  - **Important fact:**
    - Because the **life time** of the **different** ***kinds*** of **variables** is **different**:
      - **Different** ***kinds*** of **variables** are **stored** in the **computer memory** in ***different*** **ways**

        (To **maximize efficiency** !!!)

- **Program organization in main memory**
  - The **program instructions** and the **different *kinds*** of **variables** are **stored** in the **main memory** as follows:



Explanation:

- The **program instructions** are **stored**:
  - at the **beginning** of the **main memory**

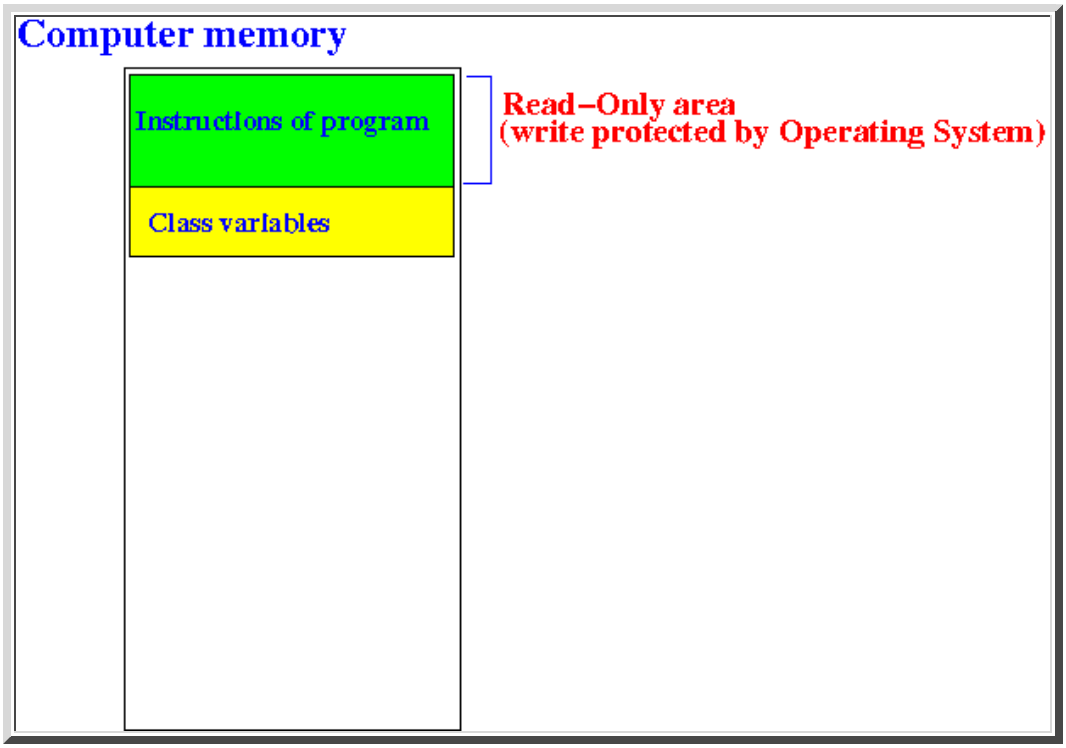
  These **memory locations** will be **marked** as **READONLY** (no writing allowed) by the **Operating System**

- The **Class variable** (which exists for the **entire program execution** are **stored** ***after*** the **program instructions**.
  - The **program instructions** and **Class variables** will continue to **occupy the memory** for **as long as** the **program is executing** !

- The **program "heap"** and **program "stack"** are:
  - ***Dynamically* changing** memory areas used to **store** ***transient* variables**

    I.e.: To **store variables** that are **created** and **destroyed** during the **execution** of the **computer program**

- **Managing the program heap**
  - When a **program starts** its **execution**, the **memory usage** will be as follows:



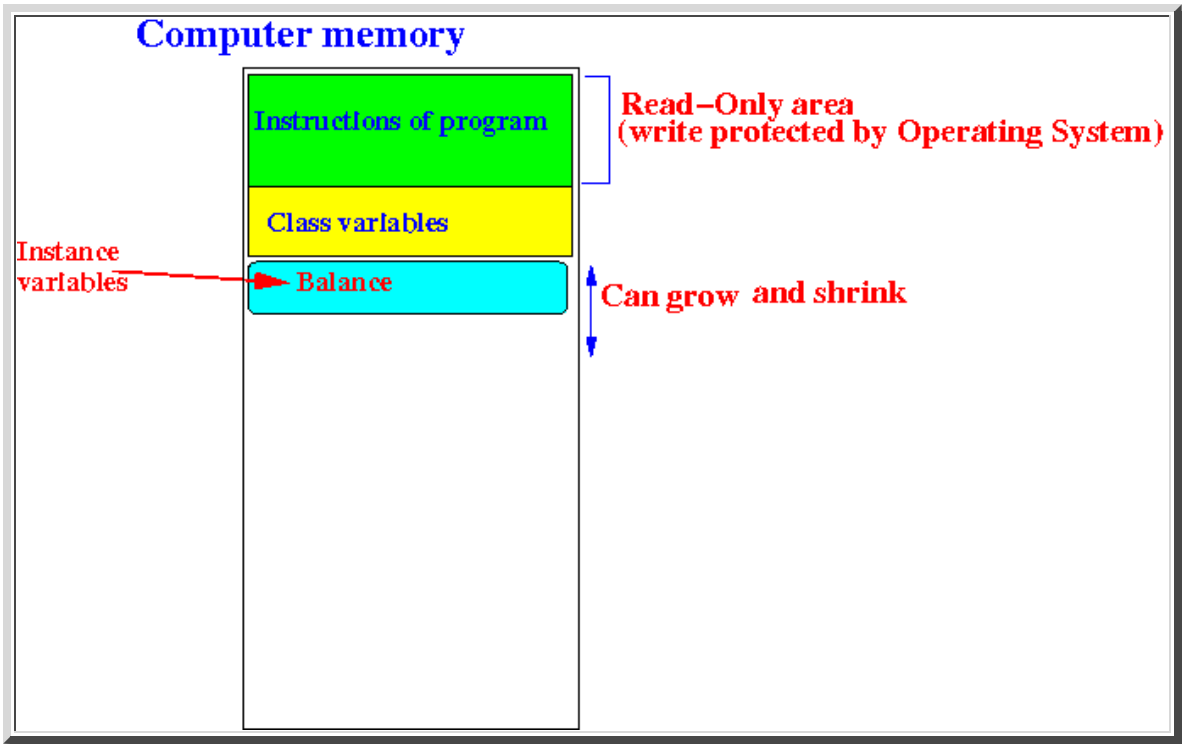
The **memory** contains **only**:

- **Program instructions**
- **Class variable**

  - When a **Java program** executes a `new` **operation**, **e.g.**:

```
x = new BankAccount( .... );
```

the `new` **operation** will **allocate (= reserve) memory** in the **program** ***heap*** for **all** the **instance variables** in the **(`BankAccount`) object**:

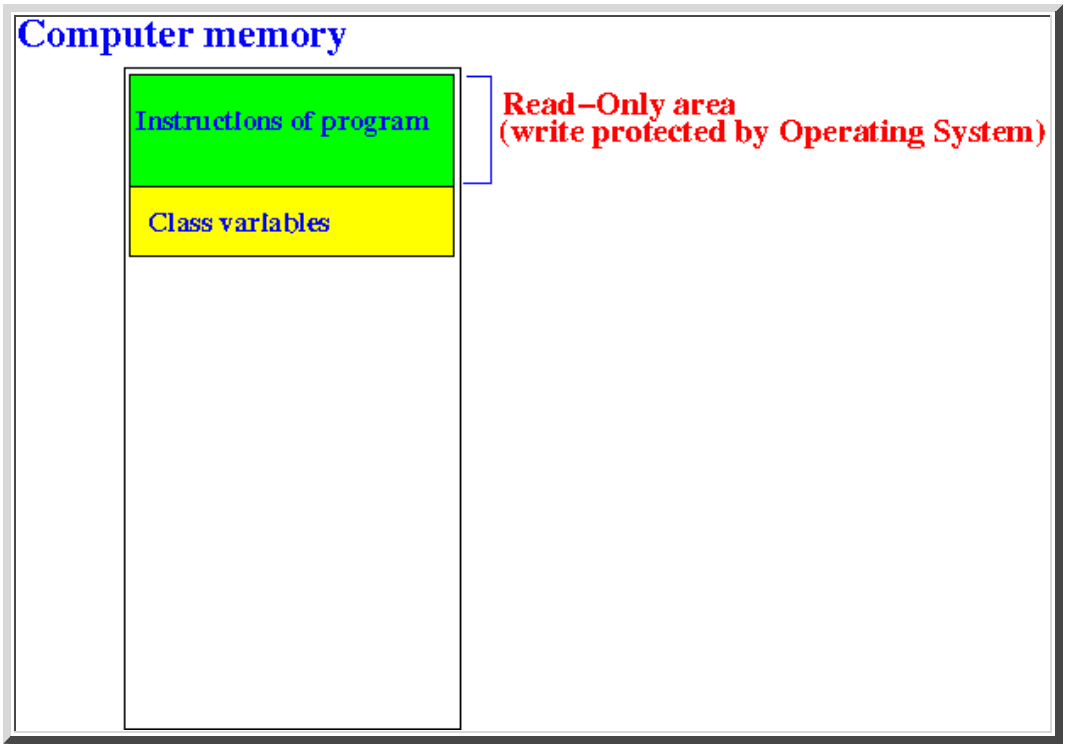


Note:

- The **Java run time system** will **maintain information** on ***which* memory locations** are:
  - ***Available* for use** (i.e., ***free***)
  - ***Occupied***

  so that a **memory cell** is **not used** for/by ***different* variables** !!!

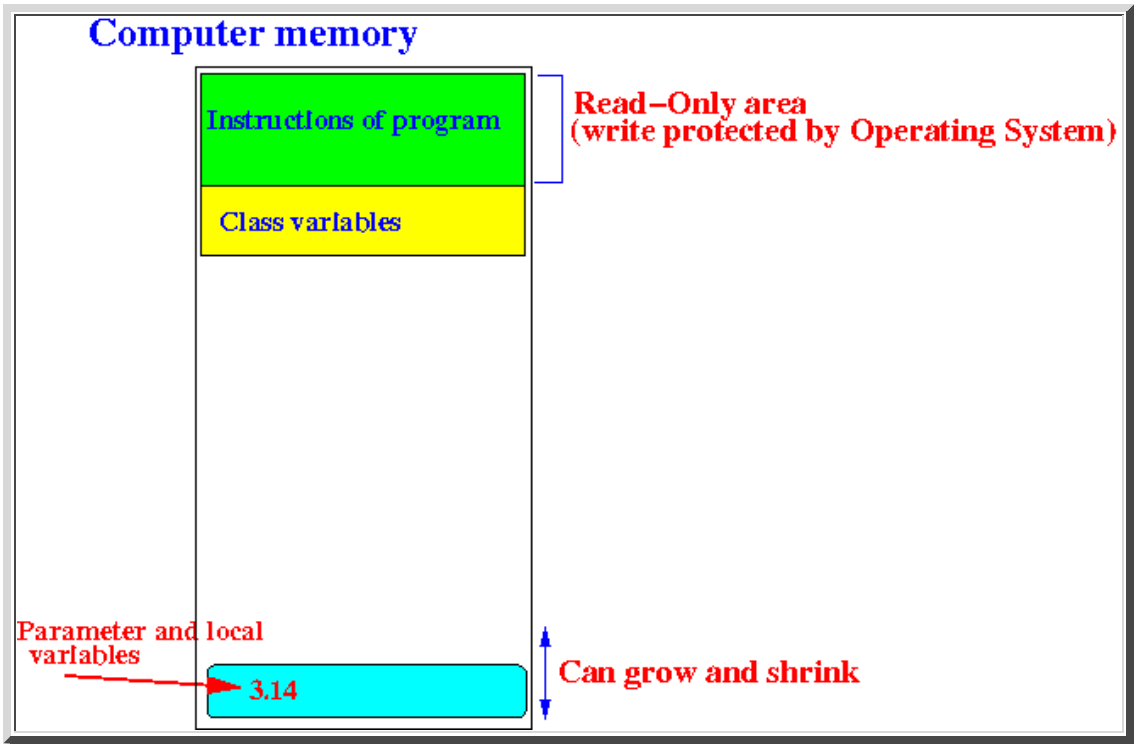
- **Managing the program stack**
  - **Restarting** the **discussion** from the **beginning**..... When a **program starts** its **execution**, the **memory usage** will be as follows:



- When we **invoke** a **method**, **e.g.:**

```
x = Math.sin( 3.14 );
```

The **method invocation** will **allocate (= reserve memory)** the **parameter variables** and the **local variables** of the **method** on the **program stack**:



- **Computer jargon: "create" a variable and "destroy" a variable**
  - **"Creating" a variable**:
    - **Create a variable** = **allocate (= reserve)** the **memory space (= cells)** for a **variable**

      (**Memory cells** that are **allocated (= reserved)** can ***not*** **be used** for another **variable** !!!)

    In other words:
    - There is **no *real* creation** taking place in **variable "creation"**

      (Nothing like the creation in Genesis 1:1 of the Bible)

  - **"Destroying" a variable**:
    - **Destroy a variable** = **de-allocate (= unreserve)** the **memory space** used by the **variable**

      (The **memory cells** used by the **variable** can now be ***re-used*** by ***another*** **variable** !!!)

- **Postscript**
  - I will **explain**
    - **How** to **allocate/de-allocate** memory in more details **later** in the course
  - **Later**, I will **also explain**:
    - **Why** we use a **stack** to **store *parameter* and *local* variables** (click here)....