SchoolNova

IT101

Object Oriented Programming

History

- In the old days, the standard programming technique was called "procedural programming." That's because the emphasis was on the procedures or tasks that made up a program.
- Developers designed their programs around what they thought were the key procedures. Typically, the entire program was written in one file.
- As computer programs grew bigger and more complex, the program files became longer, very difficult to maintain and error prone. As a result, IT professionals had to change their approach to program design.
- Today, the most popular programming technique is *object-oriented programming* (OOP).
- With OOP, instead of thinking first about functional procedures, you think first about the entities in your problem. The entities are called objects.
- The first object-oriented programming language was SmallTalk, developed in 1960-ies in Norway, but it did not become popular until 1980-ies.
- The object oriented C++ language was created by Bjarne Stroustrup of Bell Labs out of the procedural C language. Bjarne wrote the following about the differences between the two design approaches:
  - *A programming language serves two related purposes: it provides a vehicle for the programmer to specify actions to be executed and a set of concepts for the programmer to use when thinking about what can be done. The first aspect ideally requires a language that is "close to the machine", so that all important aspects of a machine are handled simply and efficiently in a way that is reasonably obvious to the programmer. The C language was primarily designed with this in mind. The second aspect ideally requires a language that is "close to the problem to be solved" so that the concepts of a solution can be expressed directly and concisely. The facilities added to C to create C++ were primarily designed with this in mind.*
- Today, Java is one of the most popular Object Oriented Programming languages.

Object

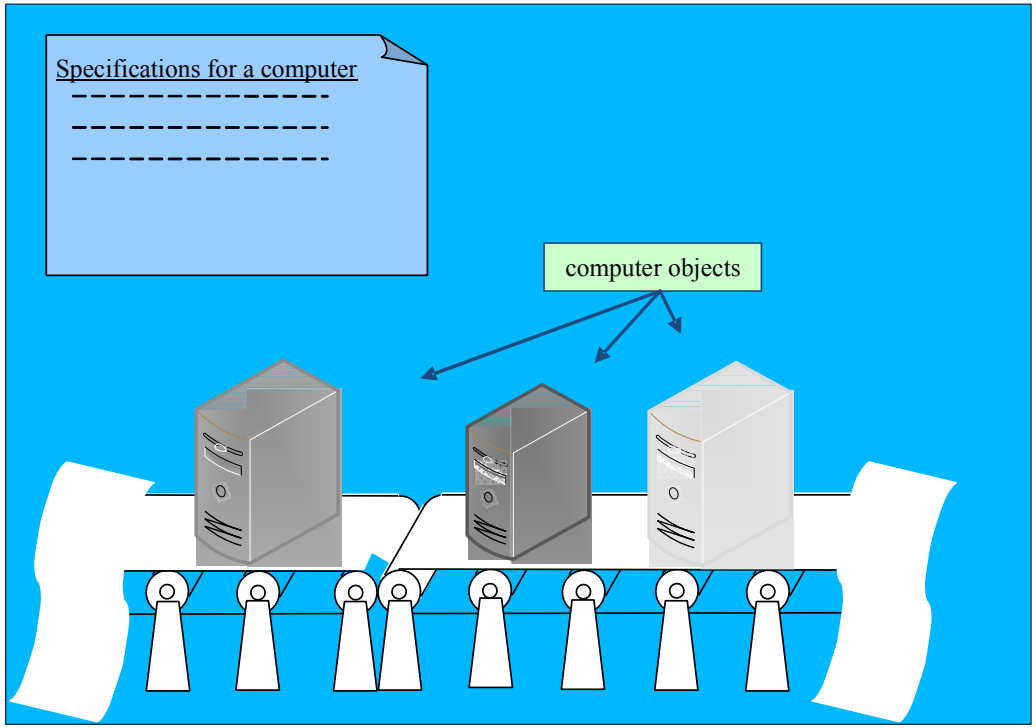
- An object is a set of related data which identifies the current state of the object and a set of its behaviors.
- Examples of objects:

human entities	physical objects	mathematical entities
employees	cars in a traffic-flow simulation	points in a coordinate system
customers	aircraft in an air-traffic control system	complex numbers
students	electrical components in a circuit-design program	time

- Benefits of OOP:
  - **Programs are more understandable** - since people tend to think about problems in terms of objects, it's easier for people to understand a program that's split into objects.
  - **Fewer errors** - since objects provide encapsulation (isolation) for the data, it's harder for the data to get messed up.

Class

- A class is a description (template or blueprint) for a set of objects.
- Note the three computers on a conveyer belt in a manufacturing plant:
  - The three computers represent objects, and the specifications document represents a class. The specifications document is a blueprint that describes the computers: it lists the computers' components and describes the computers' features.
- Think of an object as a physical example for a class's description. More formally, we say that an object is an instance of a class.



Variables and Methods

- A class's variables specify the type of data that an object can store. For example, if you have a class for computer objects, and the Computer class contains a hardDiskSize instance variable, then each computer object stores a value for the size of the computer's hard disk. Example:

```
class Computer {
  public int hardDiskSize;
}
```

- A class's methods specify the behavior that an object can exhibit. For example, if you have a class for computer objects, and the Computer class contains a printSpecifications instance method, then each computer object can print a specifications report (the specifications report shows the computer's hard disk size, CPU speed, cost, etc.). Example:

```
class Computer {
  public String printSpecifications() {
   // ...
  }
}
```

- Java methods must specify what the method returns. A method may return a primitive value, an object instance or an array. A method may also return nothing, such as the "main" method, in which case the return type is "void": public static void main (String args []);

- It is possible to pass one or more values to a method when the method is called. These values are called arguments. In the "main" method example above the method accepts an array of String objects as an argument.

Homework

- Create a new Java project with “com.presidents” package in NetBeans;
- Create a class named “President”. The class should have two public instance variables of type String: “name” and “country”.
- The class should have one public method “toString”. The method should return a String with the president's name and country, for example:

```
public String toString() {
 return this.name + " is the president of " + this.country;
}
```

- Create another class in the same package, name it “PresidentsList”.
- The PresidentsList class should have only the main method that creates an array of 5 instances of the President class for USA (Donald Trump), Russia (Vladimir Putin), China (Xi Jinping), France (Emmanuel Macron) and Mexico (Enrique Pena Nieto). Then loop through the array and output the President's toString method to the console. Example of the PresidentsList.main method:

```
public static void main (String args[]) {
  President [] pArray = new President[5];
  pArray[0]  = new President();
  pArray[0].country = "USA";
  pArray[0].name = "Donald Trump";
  // ... - repeat for the remaining four presidents
  for (int i = 0; i < pArray.length; i++) {
   System.out.println(pArray[i].toString());
  }
}
```