Von Neumann Model

Processor (CPU)

Data

Programs or Algorithms

Memory
Von Neumann Model

Processor (CPU)

Data Programs or Algorithms

Memory

Processor instructions: +, -, *, /, read, write, <, <=, >, >=, ≠, ...

Binary: 0, 1

Data Programs or Algorithms
public class PrintHello {
    
    public static void main (String[] args) {
        System.out.println("Hello");
    }
}
Binary program that prints “hello”

10111010000011000000000110110100
00001001110011010010000110111000
000000001001100110011010001010111
01001000011001010110110001101100
01101111001011000010000001010111
011011110111001001101100011001000
00100001000011010000110001101111

for processor 8086

Machine or executable code
Program in high level programming language → Compiler → equivalent program in \{ machine or executable code \\
\text{OR} \\
intermediate code \}
Python and Java compilers produce intermediate code called byte code.
Java program
Must have extension .java

Java compiler
javac

Java byte code
Has extension .class

Interpreter or virtual machine
java

execute program
public class PrintHello {

    public static void main (String[] args) {
        System.out.println("Hello");
    }
}

```c
int b, c;
int a = 1;
if (foo() == 1) b = 2;
else b = "house";  ← compilation error
```
Software Life Cycle:

- Specification
- Design (Data, Algorithms)
- Implementation (Translation of design to a programming language)
- Testing and debugging
Interface

Public part

Private part

Module

Information hiding, or Encapsulation
The diagram illustrates a bank layout with different areas identified:

- **Public** area includes:
  - Reception
  - Teller

- **Private** area includes:
  - Vault
  - Interface

The interface is connected to both the public and private areas, suggesting a connection point between the reception and teller areas and the vault.
<table>
<thead>
<tr>
<th>Private</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data (instance variables)</td>
<td>Data (instance variables)</td>
</tr>
<tr>
<td>Algorithms (methods)</td>
<td>Algorithms (methods)</td>
</tr>
</tbody>
</table>
Objects have

• **Data** *(properties of an object)*

• **Algorithms** *(or behaviours or actions)*
Objects have

- **Data** *(properties of an object)*
  - In Java they are called *attributes* or *fields* or *instance variables*

- **Algorithms** *(or behaviours or actions)*
Objects have

• **Data** (*properties of an object*)
  • In Java they are called *attributes* or *fields* or *instance variables*

• **Algorithms** (*or behaviours or actions*)
  • In Java they are implemented as *methods*
    (more specifically, *instance methods*)
Objects have

- **Data** *(properties of an object)*
  - In Java they are called *attributes* or *fields* or *instance variables*

- **Algorithms** *(or behaviours or actions)*
  - In Java they are implemented as *methods* (more specifically, *instance methods*)

Both can be private or public.
<table>
<thead>
<tr>
<th>Private</th>
<th>Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data (instance variables)</td>
<td>Data (instance variables)</td>
</tr>
<tr>
<td>Algorithms (methods)</td>
<td>Algorithms (methods)</td>
</tr>
</tbody>
</table>
Objects and Classes

Every object belongs to a specific class.

A class specifies the instance variables and methods of an object.
Objects and Classes

Every object belongs to a specific class.

A class specifies the instance variables and methods of an object, so a class definition consists of

• Instance variable declarations (also known as fields or attributes), and
• Method definitions
Example of a Java Class

```java
public class Person {
    /* Attribute declarations */
    private String lastName;
    private String firstName;
    private String email;

    ... 

    // Method declarations
    public String getEmail() {
        return email;
    }

    public void setEmail(String mail) {
        email = mail;
    }
}
```
Objects and Classes

Every object belongs to a specific class. A class specifies the instance variables and methods of an object, so a class definition consists of

- Instance variable declarations (also known as fields or attributes), and
- Method definitions

A class definition must be stored in a file with the same name as the class and a .java extension.
Data Types

Java data types:

- primitive
- non-primitive or classes
Data Types

Java data types:

- primitive
- non-primitive or classes

There are 8 primitive Java data types:

boolean, char, byte, short, int, long, float, double

8 bits 16 32 64
integer

32 64
real

2.7145
Computer Model

Processor

Memory
```plaintext
int a, b;
a = 52;
b = a + 1;
52 + 1

Person p;
p = new Person();
```

### Symbol Table

<table>
<thead>
<tr>
<th>Var</th>
<th>Type</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>int</td>
<td>10</td>
</tr>
<tr>
<td>b</td>
<td>int</td>
<td>20</td>
</tr>
<tr>
<td>p</td>
<td>Person</td>
<td>100</td>
</tr>
</tbody>
</table>

### Memory

<table>
<thead>
<tr>
<th>Address</th>
<th>8 bits = 1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0101 001</td>
</tr>
<tr>
<td>1</td>
<td>1110 111</td>
</tr>
<tr>
<td>2</td>
<td>1101 111</td>
</tr>
<tr>
<td>3</td>
<td>0001 1101</td>
</tr>
<tr>
<td></td>
<td>1101 1101</td>
</tr>
<tr>
<td></td>
<td>1010 1010</td>
</tr>
</tbody>
</table>

### Conversion

52 in binary:

```
26
252
13
2126
0
2113
```

Converting 52 to binary:

```
110100
```

### Address:

```
1101 001
```

### Memory Address:

```
100
```

### Person Object:

```
FirstName:
LastName:
getEmail:
getEmail:
```
Data Types

• A variable of a **primitive** type stores a **value**.
• A variable of a **non-primitive** type stores and **address** or **class**.
Example of a Java Class

```java
public class Person {

    /* Attribute declarations */
    private String lastName;
    private String firstName;
    private String email;

    ...

    // Method declarations
    public String getEmail() {
        return email;
    }

    public void setEmail (String mail) {
        email = mail;
    }
}
```
```java
int a, b;
a = 52;
b = a + 1;
Person p;
p = new Person();
P. setName("joe", "doe");
p = new Person("joe", "doe", "jd@uwocu");
```

### Symbol Table

<table>
<thead>
<tr>
<th>var</th>
<th>type</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>int</td>
<td>6</td>
</tr>
<tr>
<td>b</td>
<td>int</td>
<td>21</td>
</tr>
<tr>
<td>p</td>
<td>Person</td>
<td>100</td>
</tr>
<tr>
<td>s</td>
<td>String</td>
<td>200</td>
</tr>
</tbody>
</table>

De-referencing:

```java
De-referencing
```

Garbage Collector:

- Memory
- String

```java
a = 52;
b = a + 1;
p = new Person();
```

```java
Person p;
p = new Person("joe", "doe", "jd@uwocu");
```
Constructors

A constructor is a special method that is called automatically when an object is created with the new operator.

Its purpose is to initialize the attributes of an object when the object is created.

In Java a constructor has the same name as the class name.
public class Person {

    /* Attribute declarations */
    private String lastName;    // last name
    private String firstName;   // first name
    private String email;       // email address

    /* Constructor initializes the person's name and email address */
    public Person() {
        lastName = "";
        firstName = "";
        email = "";
    }

    public String getName() {
        return firstName + " " + lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail (String mail) {
        email = mail;
    }

    public void setName(String newFirst, String newLast) {
        firstName = newFirst;
        lastName = newLast;
    }
}
/* Constructor initializes the person's name and email address */

public Person() {
    lastName = "";
    firstName = "";
    email = "";
}

public Person(String first, String last, String mail) {
    firstName = first;
    lastName = last;
    email = mail;
}

Is this allowed?
/* Constructor initializes the person's name and email address */

public Person() {
    lastName = "";
    firstName = "";
    email = "";
}

public Person(String first, String last, String mail) {
    firstName = first;
    lastName = last;
    email = mail;
}

Overloading: Two or more methods have the same name
/* Constructor initializes the person's name and email address */

public Person() {
    lastName = "";
    firstName = "";
    email = "";
}

public Person(String first, String last, String mail) {
    firstName = first;
    lastName = last;
    email = mail;
}

Signature: Method name + number and types of parameters
/* Constructor initializes the person's name and email address */

public Person() {
    lastName = "";
    firstName = "";
    email = "";
}

public Person() {
    firstName = "John";
    lastName = "Doe";
    email = "jd@mail";
}

**Signature:** Method name + number and types of parameters
public class Person {

    /* Attribute declarations */
    private String lastName;       // last name
    private String firstName;      // first name
    private String email;          // email address

    /* Constructor initializes the person's name and email address */
    public Person() {
        lastName = "";
        firstName = "";
        email = "";
    }

    public Person(String first, String last, String mail) {
        firstName = first;
        lastName = last;
        email = mail;
    }

    public String getName() {
        return firstName + " " + lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String mail) {
        email = mail;
    }

    public void setName(String newFirst, String newLast) {
        firstName = newFirst;
        lastName = newLast;
    }
}
Person p;
  p.getEmail();    // Would throw a Null pointer exception
p = new Person("joe", "doe", "jd@uwoc.ca");
p.getEmail();

Symbol Table

<table>
<thead>
<tr>
<th>Var</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>Person</td>
<td>100</td>
</tr>
</tbody>
</table>

p: NULL

1000
1200
1400
1600
setEmail();
getEmail();

1200
"joe"

1400
"doe"

1600
"jd@uwoc.ca"
Person p, q:
int i, j;
boolean equal;
i = 1;
j = 1;
if (i == j) equal = true;
else equal = false;
p = new Person("joe", "doe", "jd@uwo.ca");
q = new Person("joe", "doe", "jd@uwo.ca");
if (p == q) equal = true;
else equal = false;

Symbol Table

<table>
<thead>
<tr>
<th>Var</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>Person</td>
<td>100</td>
</tr>
<tr>
<td>q</td>
<td>Person</td>
<td>200</td>
</tr>
<tr>
<td>i</td>
<td>int</td>
<td>300</td>
</tr>
<tr>
<td>j</td>
<td>boolean</td>
<td>400</td>
</tr>
<tr>
<td>equal</td>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>
String s, t;
s = new String("hello");
t = "hello";
if (s.equals(t))
    System.out.println("equal");
else
    System.out.println("different");
String s, t;

s = "hello";
t = "hello";

if (s == t)
    System.out.println("equal");
else
    System.out.println("different");

<table>
<thead>
<tr>
<th>Var</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>String</td>
<td>50</td>
</tr>
<tr>
<td>t</td>
<td>String</td>
<td>100</td>
</tr>
</tbody>
</table>

Memory

- s: 
  - Address: 500
  - Value: "hello"

- t: 
  - Address: 100
  - Value: "hello"
Person $p$, $q$.  

```java
int i, j;
boolean equal;
i = 1;
j = 1;
if (i == j) equal = true;
else equal = false;
p = new Person("joe", "doe", "jd@uwo.ca");
q = new Person("joe", "doe", "jd@uwo.ca");
if ($p == q$) equal = true;
else equal = false;
```

---

<table>
<thead>
<tr>
<th>Var</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>Person</td>
<td>100</td>
</tr>
<tr>
<td>$q$</td>
<td>Person</td>
<td>200</td>
</tr>
<tr>
<td>$i$</td>
<td>int</td>
<td>300</td>
</tr>
<tr>
<td>$j$</td>
<td>int</td>
<td>400</td>
</tr>
<tr>
<td>Equal</td>
<td>boolean</td>
<td>500</td>
</tr>
</tbody>
</table>
public class Person {

    /* Attribute declarations */
    private String lastName;    // last name
    private String firstName;   // first name
    private String email;       // email address

    /* Constructor initializes the person's name and email address */
    public Person() {
        lastName = "";
        firstName = "";
        email = "";
    }

    public String getName() {
        return firstName + " " + lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String mail) {
        email = mail;
    }

    public void setName(String newFirst, String newLast) {
        firstName = newFirst;
        lastName = newLast;
    }
}
/**
 * equals method determines whether two Person objects have the same name
 * @param other: Person object that this is compared to
 * @return true if they have the same first name and last name, false otherwise
 */

public boolean equals(Person other) {
    if (this.firstName.equals(other.firstName) && this.lastName.equals(other.lastName))
        return true;
    else
        return false;
}
/**
 * equals method determines whether two Person objects
 * have the same name
 * @param other: Person object that this is compared to
 * @return true if they have the same first name and last
 * name, false otherwise
 */

public boolean equals(Person other)
{
    if (this.firstName.equals(other.firstName) &&
        this.lastName.equals(other.lastName))
        return true;
    else
        return false;
}
/**
 * equals method determines whether two Person objects have the same name
 * @param other: Person object that this is compared to
 * @return true of they have the same first name and last name, false otherwise
 */

public boolean equals(Person other){
    if (this.firstName.equals(other.firstName) &&
        this.lastName.equals(other.lastName))
        return true;
    else
        return false;
}
/**
 * equals method determines whether two Person objects
 * have the same name
 * @param other: Person object that this is compared to
 * @return true of they have the same first name and last
 * name, false otherwise
 */

public boolean equals(Person other) {
    if (this.firstName.equals(other.firstName) &&
        this.lastName.equals(other.lastName))
        return true;
    else
        return false;
}
/**
 * toString method returns a string representation of the person
 * @return string with first name and last name, email address
 */

public String toString() {
    String s = firstName + " " + lastName + "\t" + email;
    return s;
}
}