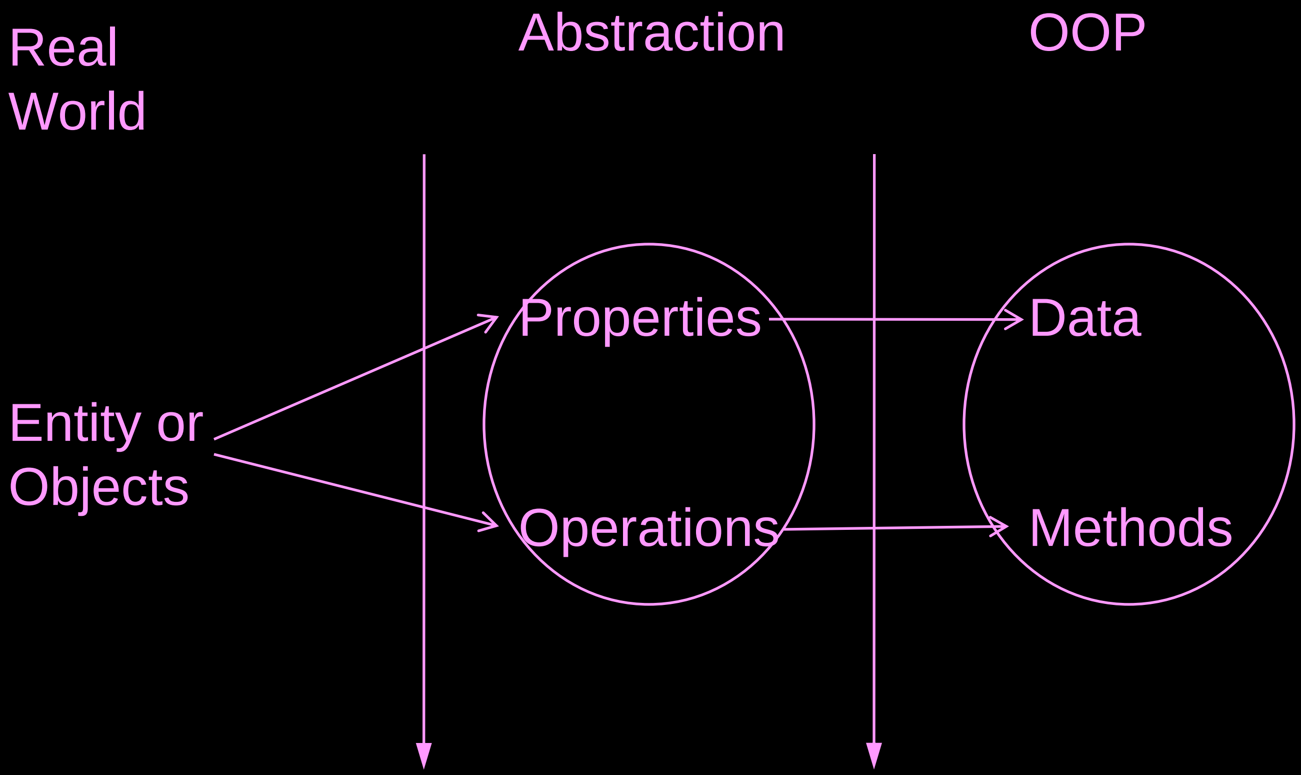
Nature of Objects and Classes

Objects


Real World
Abstraction
OOP
Entity or Objects
Properties
Operations
Data
Methods

Objects

An object is a thing.

Categories of objects:

- Tangible Things
- Roles
- Incidents
- Interactions
- Specification

Objects

Types of objects:

- Entity Objects
- Interface Objects
- Control Objects

Objects

Every object has its own

- Id
- State
- Behavior

Two object may have same state and behaviour, but different identities

Id

- Objects are identified and distinguished from one another through their identities

- At a given point of time during execution, two objects may have the same state and the same behavior, but they are distinguishable through their identities

- Can objects with nil state and nil behavior exist?

State

- Each object has its own set of local variables
- The values of these variables represents the current state of the object
- Can objects with nil state but non-nil behavior exist?

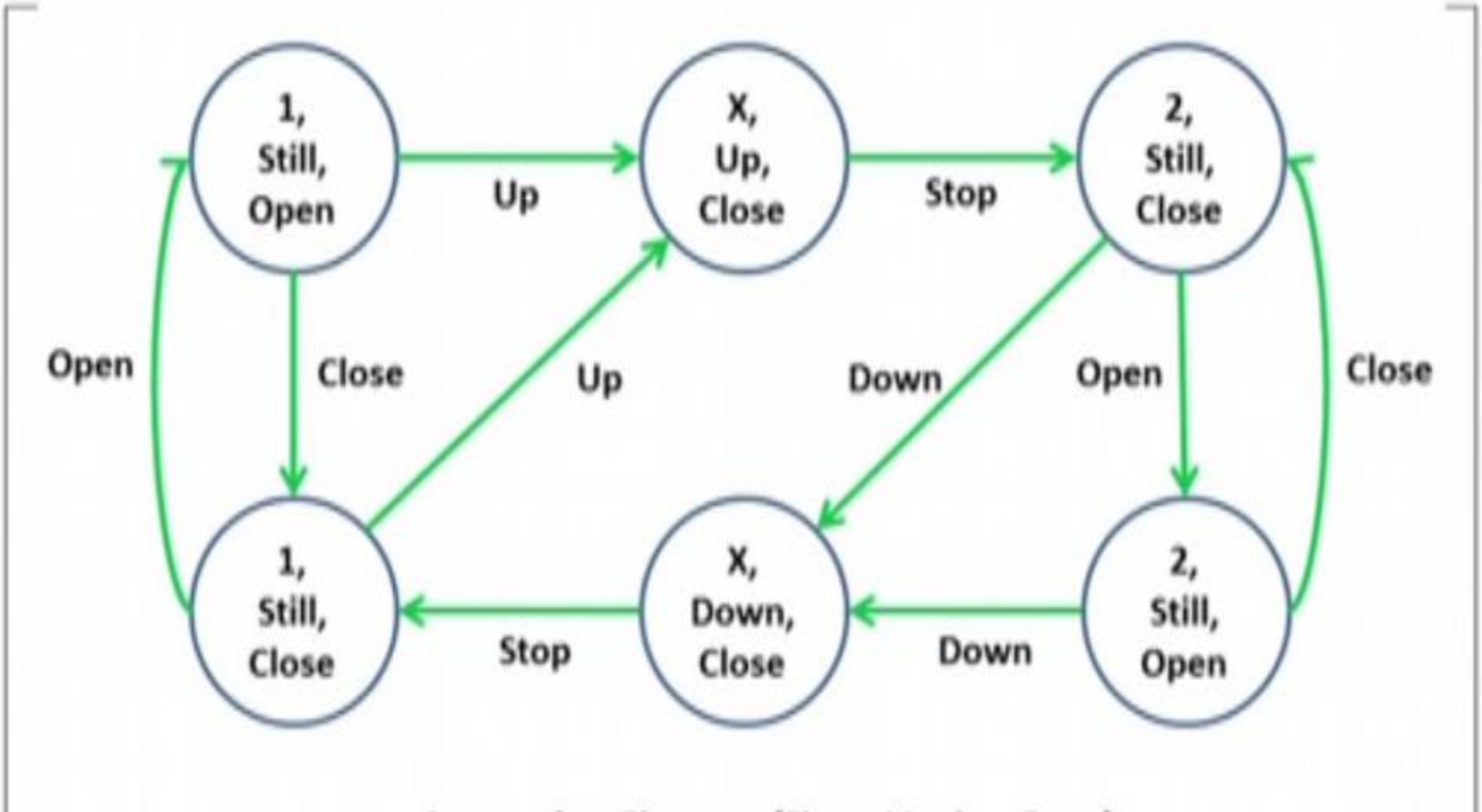
Attributes

An attribute is an abstraction of a single characteristic possessed by all the objects.

Characteristics of attributes:

- Complete
- Fully factored
- Mutually Independent

An elevator states



States of an Elevator: {Floor, Moving, Door}

Behavior

- How does an object undergo state changes?
- Member functions define the behavior
- Objects with nil behavior (no member functions) but non-nil state?

Example

```
Account
--------------------------------------------------
-Id:int
-Balance:double
-annualInterestRate:double
-dateCreated:Date
--------------------------------------------------
Account ()
Account(newId:int, newBalance:double)
Account(newId:int, newBalance: double, newAnnualInterestRate: double )
+ getId(): int
+ getBalance(): double
+ getAnnualInterestRate(): double
+ setId(newId:int)
+ setBalance(newBalance:double)
+setAnnualInterestRate(newAnnualInterestRate:double)
+setDateCreated(newDateCreated:Date)
getMonthlyInterestRate():double
withdraw(amount:double):double
Deposit( amount:double):double
```

Cont...

```
import java.util.Date;
class Account {
private int id;
private double balance;
private double annualInterestRate
private Date dateCreated;
Account () {
    id = 0;
    balance = 0.0;
    annualInterestRate = 0.0;
}
```

Cont...

```
Account(int newId, double newBalance) {
    id = newId;
    balance = newBalance;
}
Account(int newId, double newBalance, double
   newAnnualInterestRate) {
    id = newId;
    balance = newBalance;
    annualInterestRate =
   newAnnualInterestRate;
}
```

Cont…

```
public int getId() {
    return id;
}
public double getBalance() {
    return balance;
}
public double getAnnualInterestRate() {
    return annualInterestRate;
}
public void setId(int newId) {
    id = newId;
}
```

Cont…

```
public void setBalance(double newBalance) {
    balance = newBalance;
}
public void setAnnualInterestRate(double
   newAnnualInterestRate) {
    annualInterestRate =
   newAnnualInterestRate;
}
public void setDateCreated(Date
   newDateCreated) {
    dateCreated = newDateCreated;
}
```

Cont…

```
double getMonthlyInterestRate() {
    return annualInterestRate/12;
}
double withdraw(double amount) {
    return balance -= amount;
}
double deposit(double amount) {
    return balance += amount;
}
}
```

Cont…

```
public class Assign {
public static void main(String[] args) {
Account account1 = new Account(1122,
   20000.0, .045);
account1.withdraw(2500);
account1.deposit(3000);
java.util.Date dateCreated = new
   java.util.Date();
```

Cont…

```
System.out.println("Date Created:" +
   dateCreated);
System.out.println("Account ID:" +
   account1.getId());
System.out.println("Balance:" +
   account1.getBalance());
System.out.println("Interest Rate:" +
   account1.getAnnualInterestRate());
System.out.println("Balance after withdraw of
   2500:" + account1.getAnnualInterestRate());
```

Cont...

```
System.out.println("Balance after deposit of
   3000:" + account1.getAnnualInterestRate());
System.out.println("Monthly Interest:" +
   account1.getId());
System.out.println("Process completed.");
}}
```

The Nature of a Class

- In real world, many objects share common structure and behavior
  - i.e. they are of the same *kind*
  - for e.g., bicycle is a class and your black colored bicycle is an particular instance of that class
- A class is a set of objects that share a common structure and a common behavior
  - All bicycles have a state (color, speed, gear) and behavior (changeGear, stop)
  - But, different bicycles may have different colors
  - ie, for objects of a particular kind or type, they share the same *structure* and *behavior* but differ in *state*

Classes

- A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain kind

- Many objects can be instantiated from a class.

- we can have one bicycle class and many instances of that all with different colors, speed etc

Example

```
public class Bicycle{
  private int mColor;
  private int mGear;
  private double speed;
  private String manufacturer;
  public static averageCost;
  public changeGear(int whichGear) {
      . . .
  }

  public accelerate(double acceleration) {
      . . .
  }
}
```

Using classes

- After creating a class definition, we must *instantiate* it in order to use it
- By creating an instance of a class, we create an object of that type and the system allocates memory for the instance variables declared by the class
- The object's instance methods can now be invoked to make it do something