Announcements

- Midterm:
  - Date: Monday, March 1st
  - Time: 6:00pm - 8:00pm
  - Location: MAASS 112
- Review tutorials: check website for times
- Office hours

Example

```
public class Movie
{
  String title, director;

  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void print()
  {
    System.out.println(title);
    System.out.println(director);
  }
}
```

Example

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1;
    m1.print();
  }
}
```

Create objects before sending messages

```
public class MovieApplication2 {
  public static void main(String[] args)
  {
    Movie m1;
    m1.print(); // Error! Null pointer exception
  }
}
```

Create objects before sending messages

```
public class MovieApplication2 {
  public static void main(String[] args)
  {
    Movie m1;
    m1 = new Movie(“Les Invasions barbares”,
                   “Denys Arcand”);
    m1.print(); // OK
  }
}
```

Create objects before sending messages

```
public class Theater {
  void play(Movie m)
  {
    m.print();
  }
}

public class MovieApplication3 {
  public static void main(String[] args)
  {
    Movie m1;
    Theater t = new Theater();
    t.play(m1); // Error! Null pointer exception
  }
}
```

Create objects before sending messages

```
public class MovieApplication3 {
  public static void main(String[] args)
  {
    Movie m1;
    Theater t = new Theater();
    m1 = new Movie(“Les Invasions barbares”,
                   “Denys Arcand”);
    t.play(m1); // OK
  }
}
```

Example

```
public class MovieApplication4 {
  public static void main(String[] args)
  {
    Movie m1;
    m1 = new Movie(“Les Invasions barbares”,
                   “Denys Arcand”);
    String t;
    t = m1.getTitle();
  }
}
```

Methods must be defined

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void print()
  {
    System.out.println(title);
    System.out.println(director);
  }
}
```

Methods must be defined

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void print()
  {
    System.out.println(title);
    System.out.println(director);
  }
  String getTitle() { return title; }
}
```

Example

```
public class MovieApplication5 {
  public static void main(String[] args)
  {
    Movie m1;
    m1 = new Movie(“Les Invasions barbares”,
                   “Denys Arcand”);
    float t;
    t = m1.getTitle(1729);
  }
}
```

Types must match

```
public class MovieApplication5 {
  public static void main(String[] args)
  {
    Movie m1;
    m1 = new Movie(“Les Invasions barbares”,
                   “Denys Arcand”);
    String t;
    t = m1.getTitle();
  }
}
```

Types must match

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void print()
  {
    System.out.println(title);
    System.out.println(director);
  }
  String getTitle() { return title; }

  int doSomething(float x, boolean b) {
    //...
  }
}
```

Types must match

```
public class MovieApplication5 {
  public static void main(String[] args)
  {
    Movie m1;
    m1 = new Movie(“Les Invasions barbares”,
                   “Denys Arcand”);
    String t;
    t = m1.getTitle();
    m1.doSomething(“hello”);  //Error
  }
}
```

Types must match

```
public class MovieApplication5 {
  public static void main(String[] args)
  {
    Movie m1;
    m1 = new Movie(“Les Invasions barbares”,
                   “Denys Arcand”);
    String t;
    t = m1.getTitle();
    int w;
    w = m1.doSomething(1.618f, false);  //OK
  }
}
```

Types must match

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void print()
  {
    System.out.println(title);
    System.out.println(director);
  }
  String getTitle() { return title; }

  int doSomething(float x, boolean b) {
    //...
  }
  void boo(int n) {
    // ...
  }
}
```

Types must match

```
public class MovieApplication5 {
  public static void main(String[] args)
  {
    Movie m1;
    m1 = new Movie(“Les Invasions barbares”,
                   “Denys Arcand”);
    String t;
    t = m1.getTitle();

    m1.boo(m1.doSomething(1.618f, false));  //OK
  }
}
```

The "this" reference

- The reference "this" is a reserved word
- It can occur inside a normal (non-static) method
- It has a reference to the object receiving the message

Example

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void print()
  {
    System.out.println(title);
    System.out.println(director);
  }
}
```

Is the same as...

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

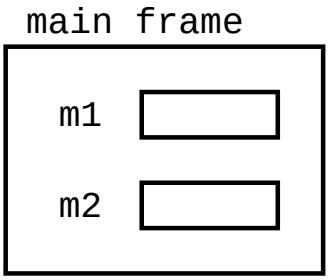
Example

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1, m2;
    m1 = new Movie(“Trainspotting”, “Danny Boyle”)
    m2 = new Movie(“Pirates of the Caribbean”,
                   “Gore Verbinski”);
    m1.print();
    m2.print();
  }
}
```

Execution

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1, m2;
    m1 = new Movie(“Trainspotting”, “Danny Boyle”)
    m2 = new Movie(“Pirates of the Caribbean”,
                   “Gore Verbinski”);
    m1.print();
    m2.print();
  }
}
```

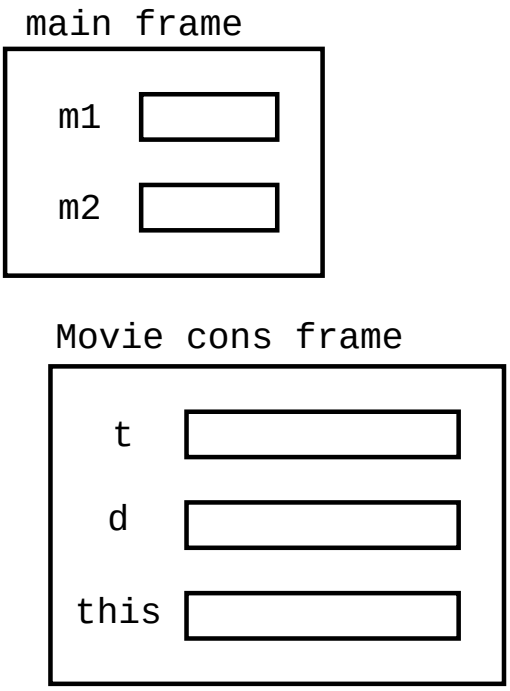
Execution



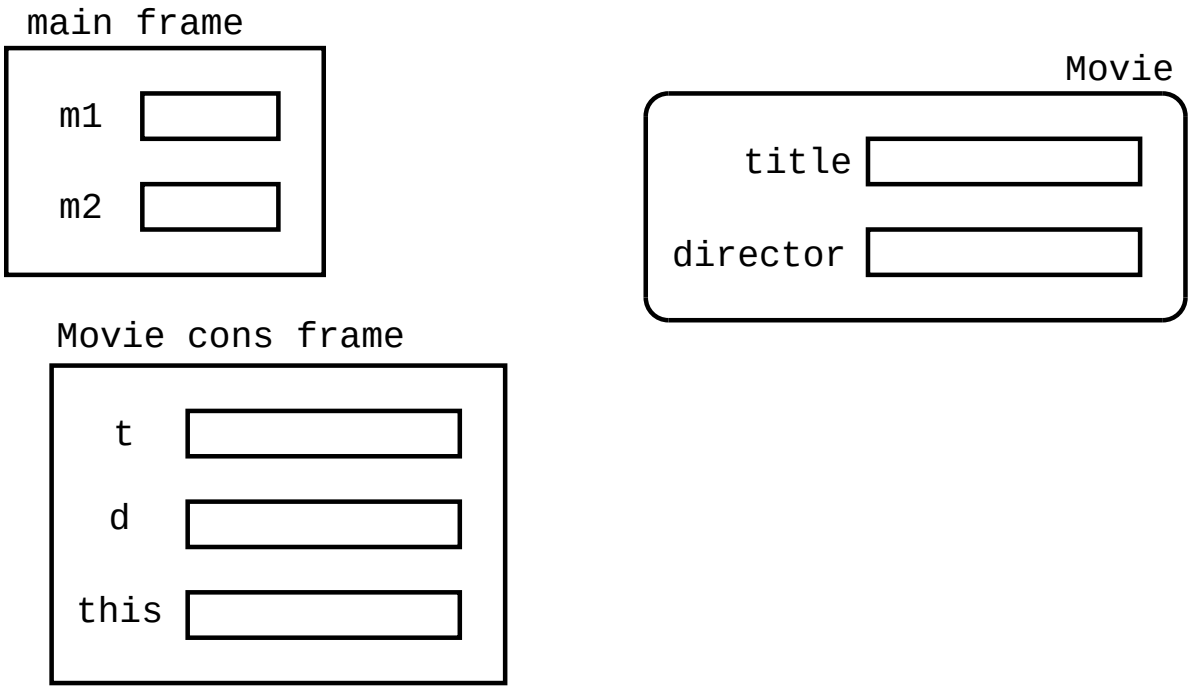
Execution

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1, m2;
    m1 = new Movie(“Trainspotting”, “Danny Boyle”)
    m2 = new Movie(“Pirates of the Caribbean”,
                   “Gore Verbinski”);
    m1.print();
    m2.print();
  }
}
```

Execution



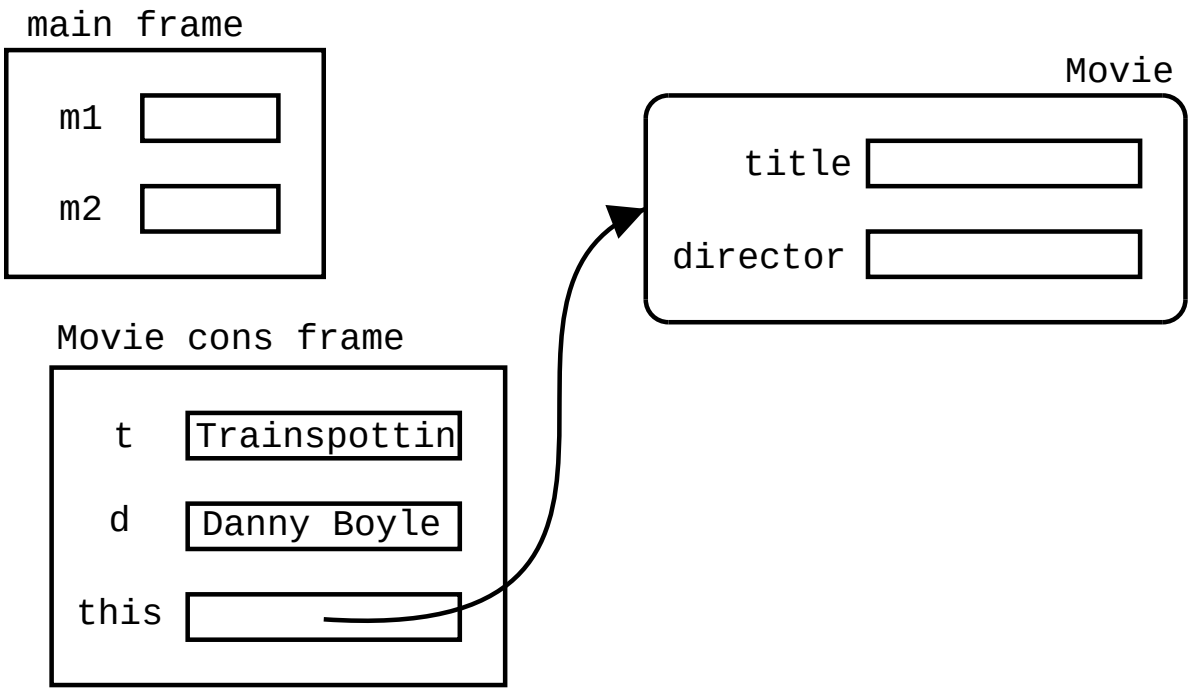
Execution


main frame
m1
m2
Movie
title
director
Movie cons frame
t
d
this

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

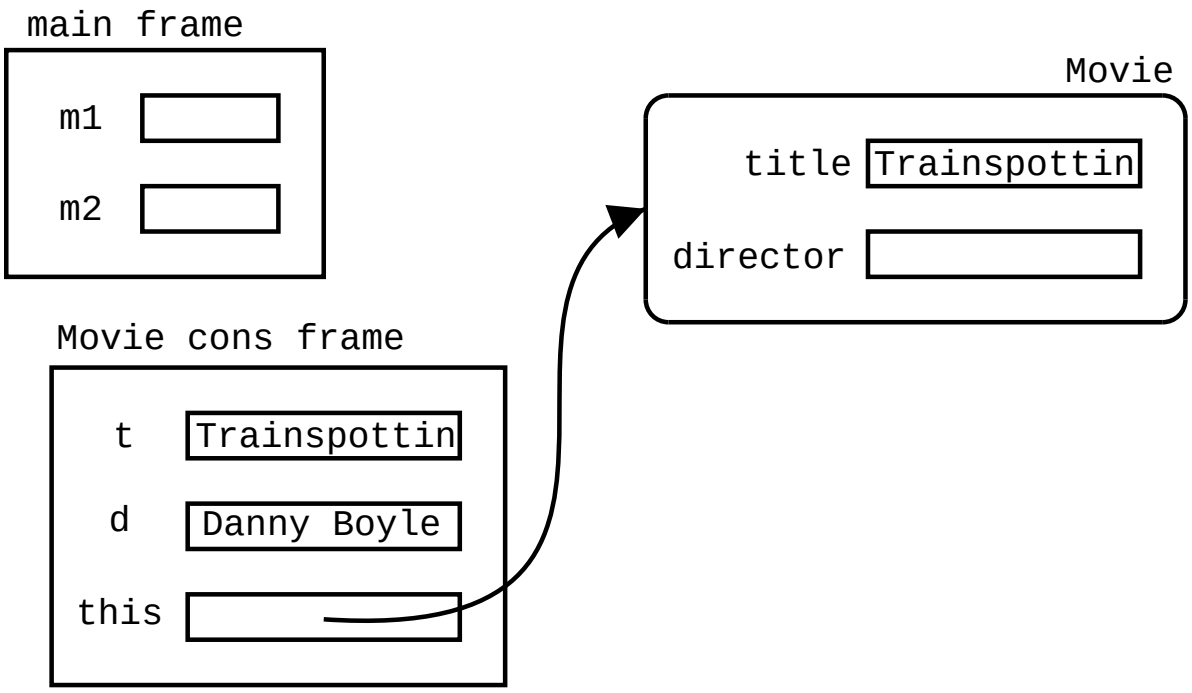
Execution


main frame
m1
m2
Movie
title
director
Movie cons frame
t
Trainspottin
d
Danny Boyle
this

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

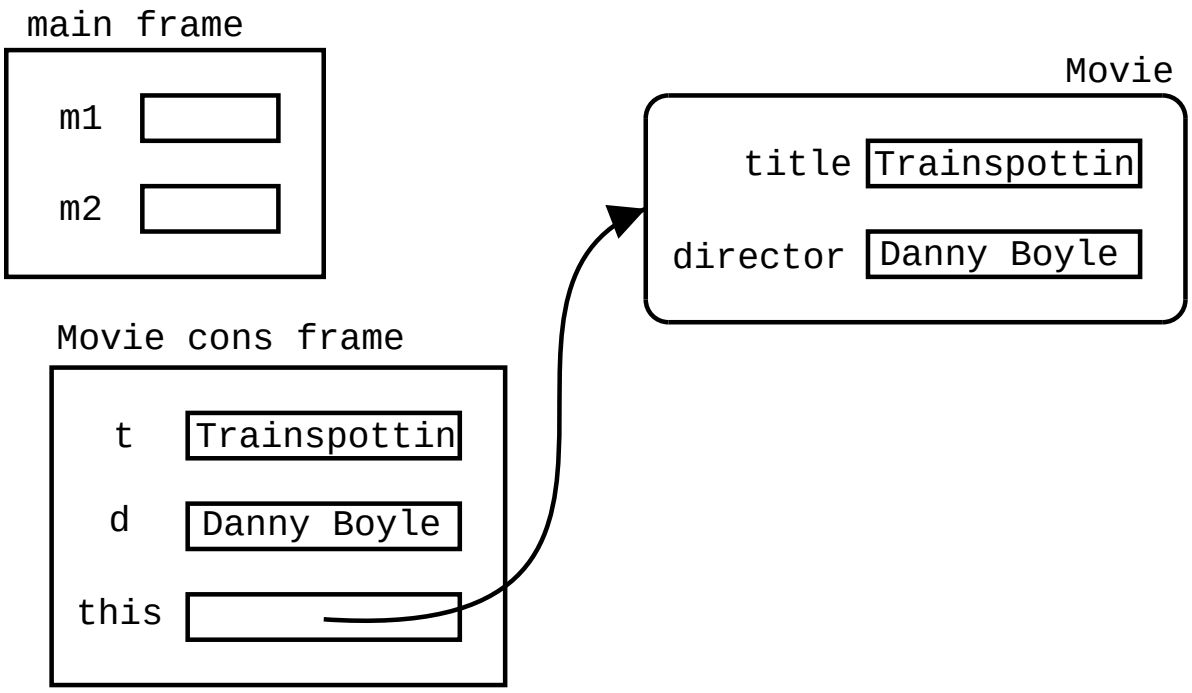
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Movie cons frame
t
Trainspottin
d
Danny Boyle
this

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

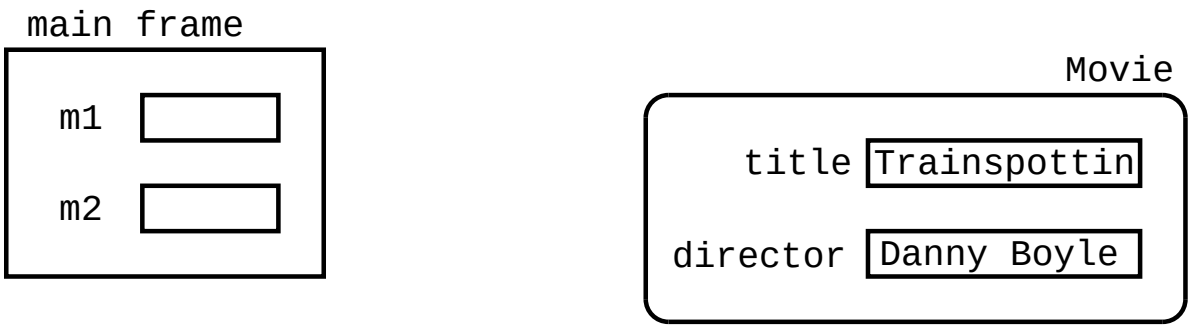
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie cons frame
t
Trainspottin
d
Danny Boyle
this

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

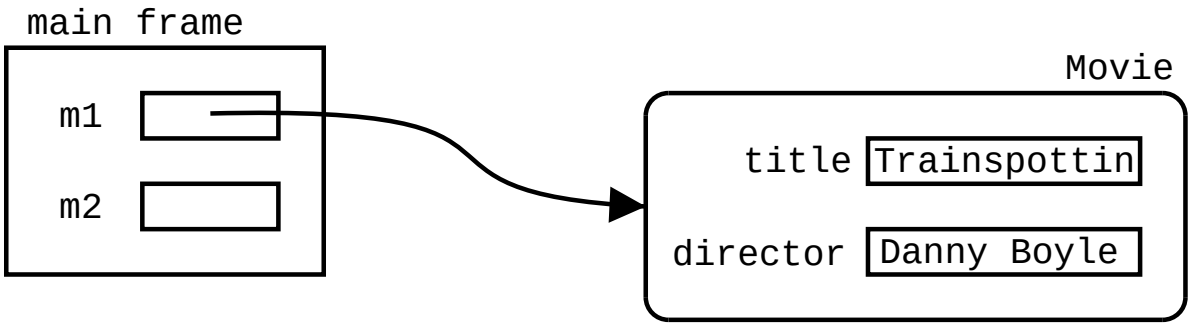
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle

Execution

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1, m2;
    m1 = new Movie(“Trainspotting”, “Danny Boyle”)
    m2 = new Movie(“Pirates of the Caribbean”,
                   “Gore Verbinski”);
    m1.print();
    m2.print();
  }
}
```

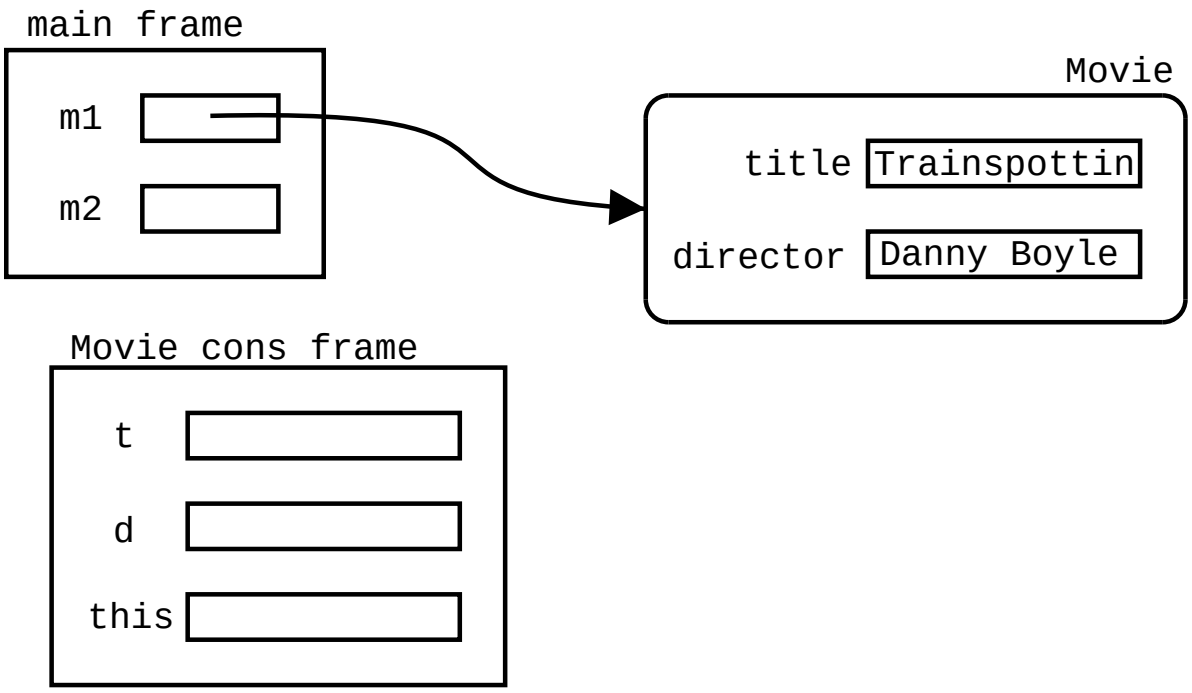
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle

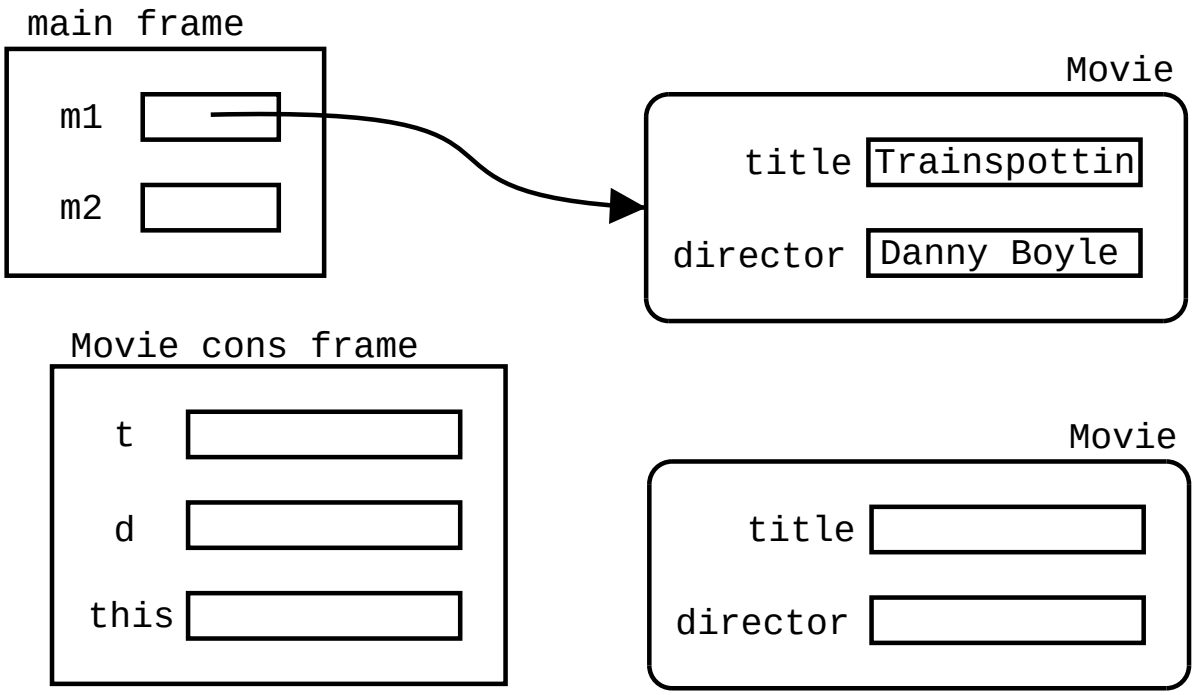
Execution

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1, m2;
    m1 = new Movie(“Trainspotting”, “Danny Boyle”)
    m2 = new Movie(“Pirates of the Caribbean”,
                   “Gore Verbinski”);
    m1.print();
    m2.print();
  }
}
```

Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie cons frame
t
d
this

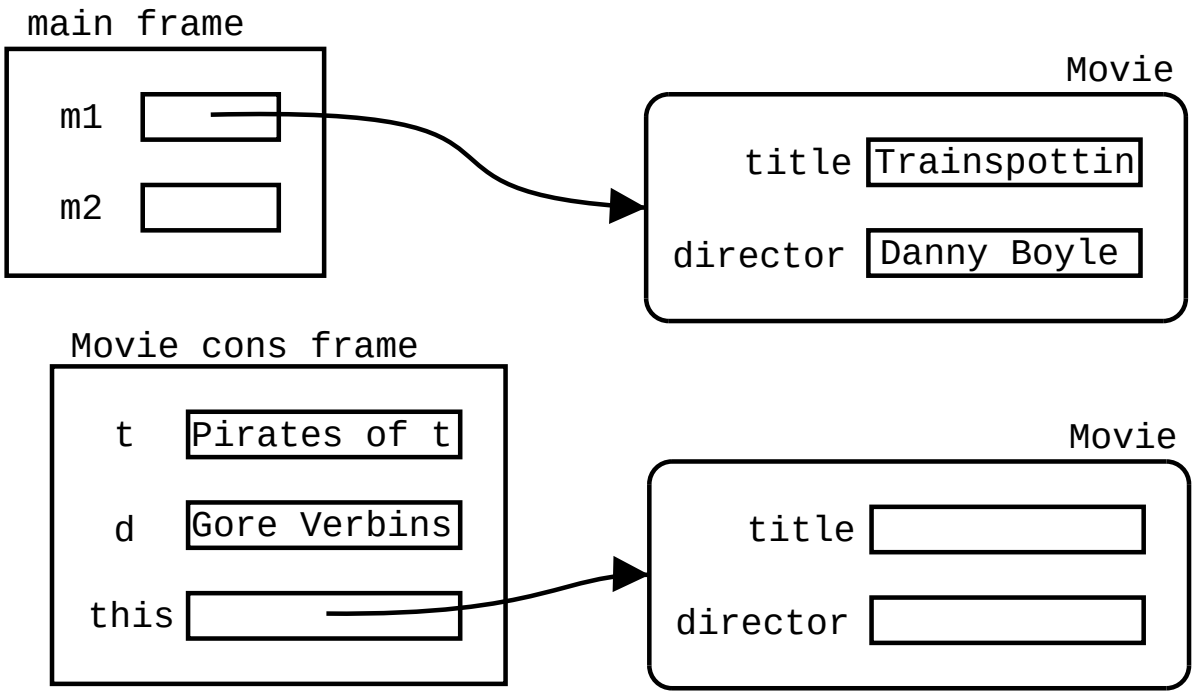
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie cons frame
t
d
this
Movie
title
director

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

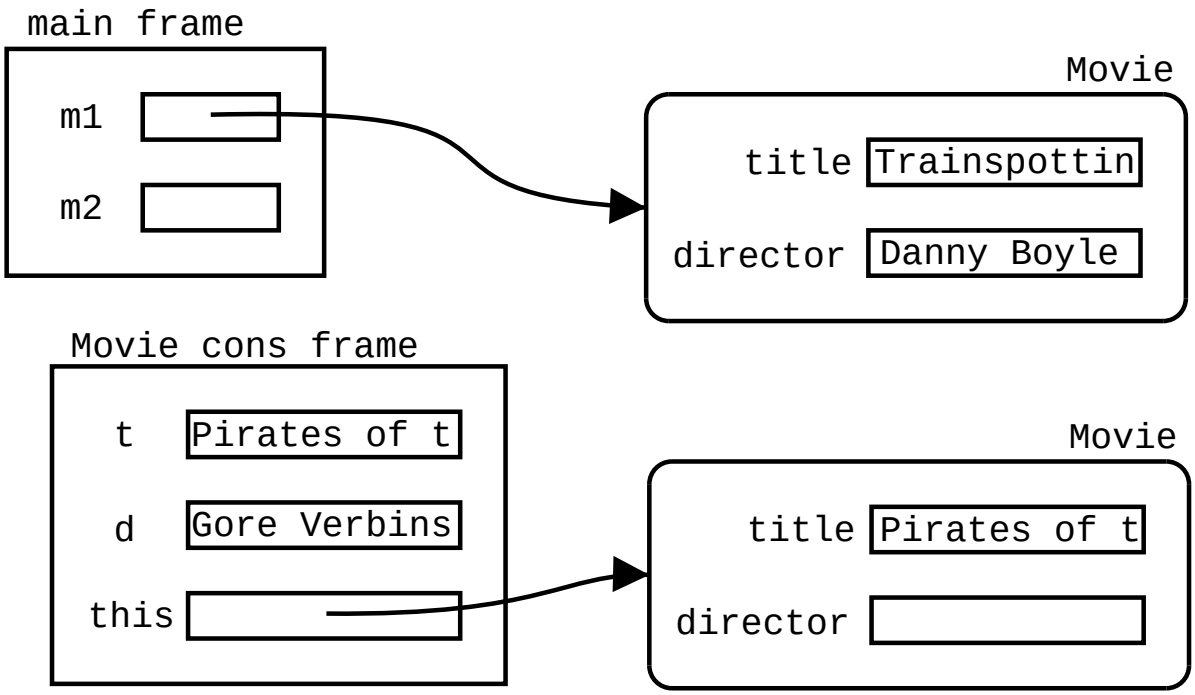
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie cons frame
t
Pirates of t
d
Gore Verbins
this
Movie
title
director

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

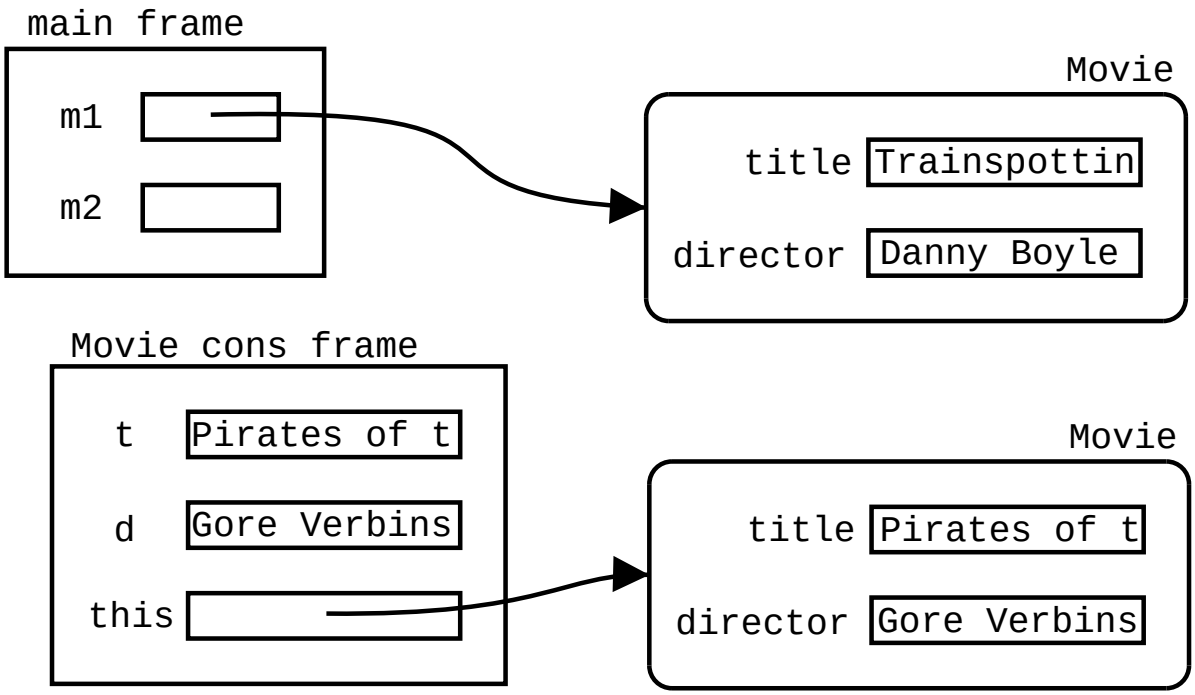
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie cons frame
t
Pirates of t
d
Gore Verbins
this
Movie
title
Pirates of t
director

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

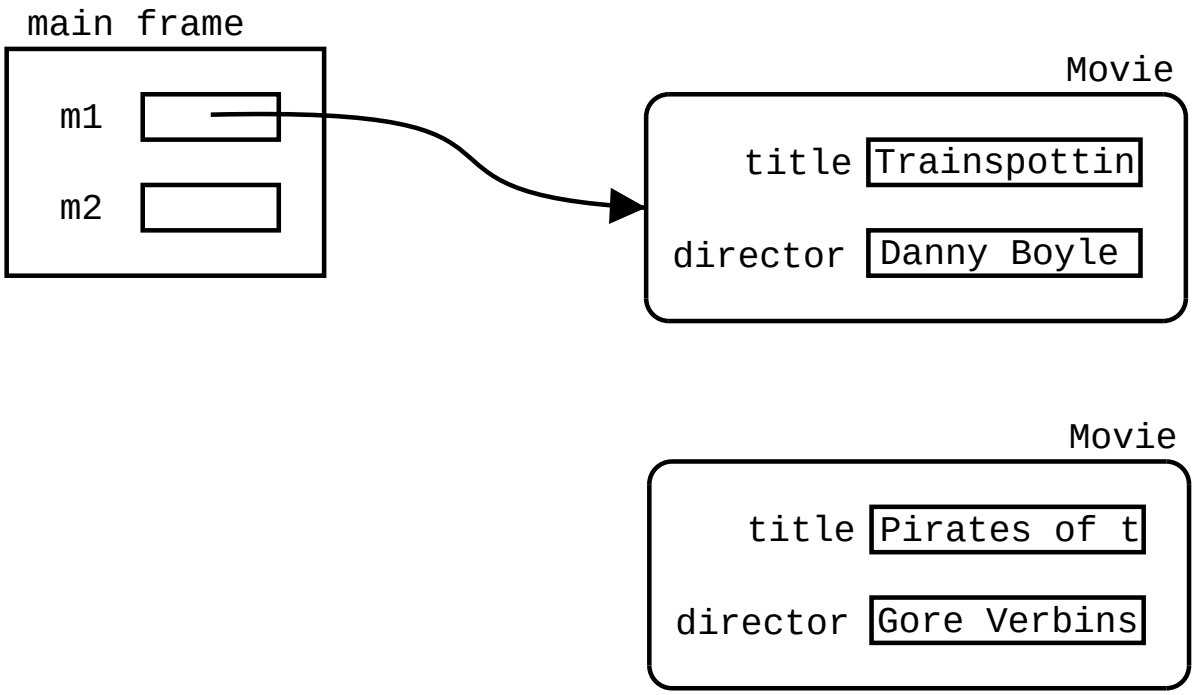
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie cons frame
t
Pirates of t
d
Gore Verbins
this
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

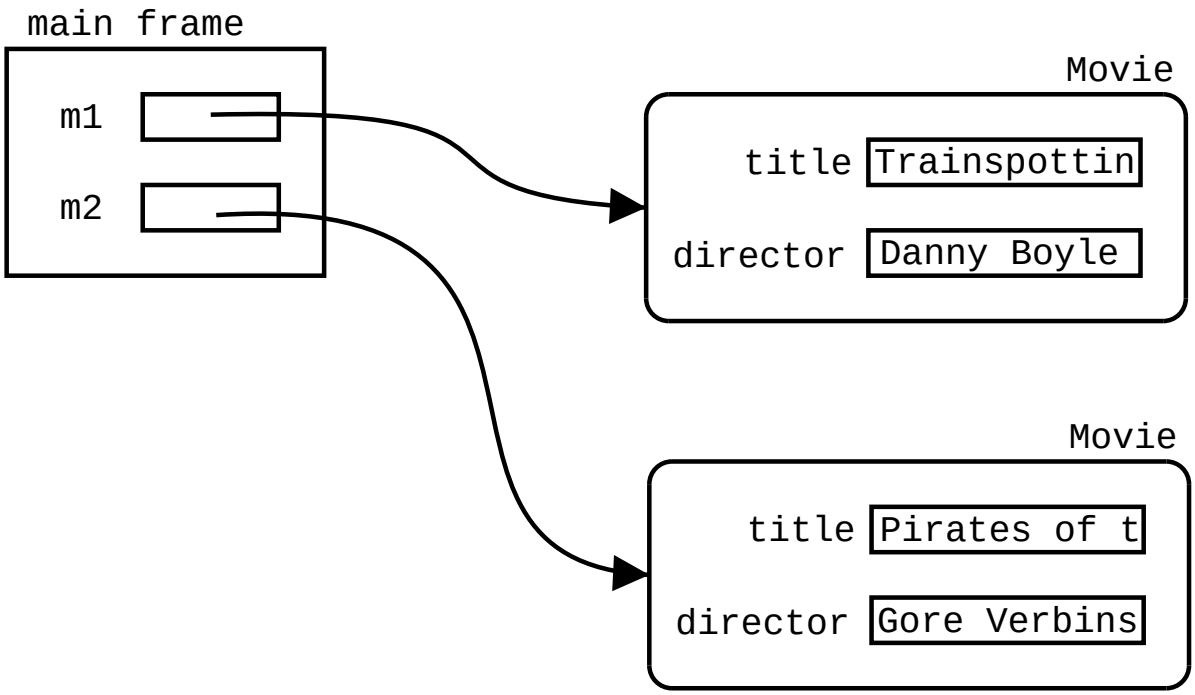
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1, m2;
    m1 = new Movie(“Trainspotting”, “Danny Boyle”)
    m2 = new Movie(“Pirates of the Caribbean”,
                   “Gore Verbinski”);
    m1.print();
    m2.print();
  }
}
```

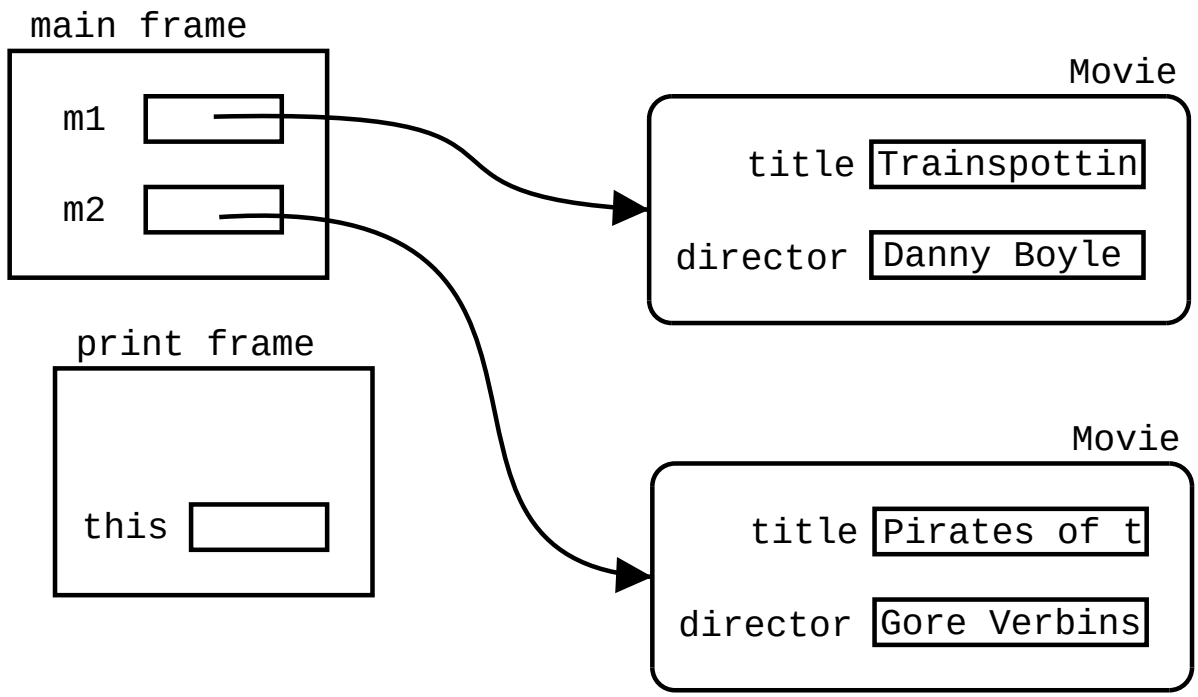
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1, m2;
    m1 = new Movie(“Trainspotting”, “Danny Boyle”)
    m2 = new Movie(“Pirates of the Caribbean”,
                   “Gore Verbinski”);
    m1.print();
    m2.print();
  }
}
```

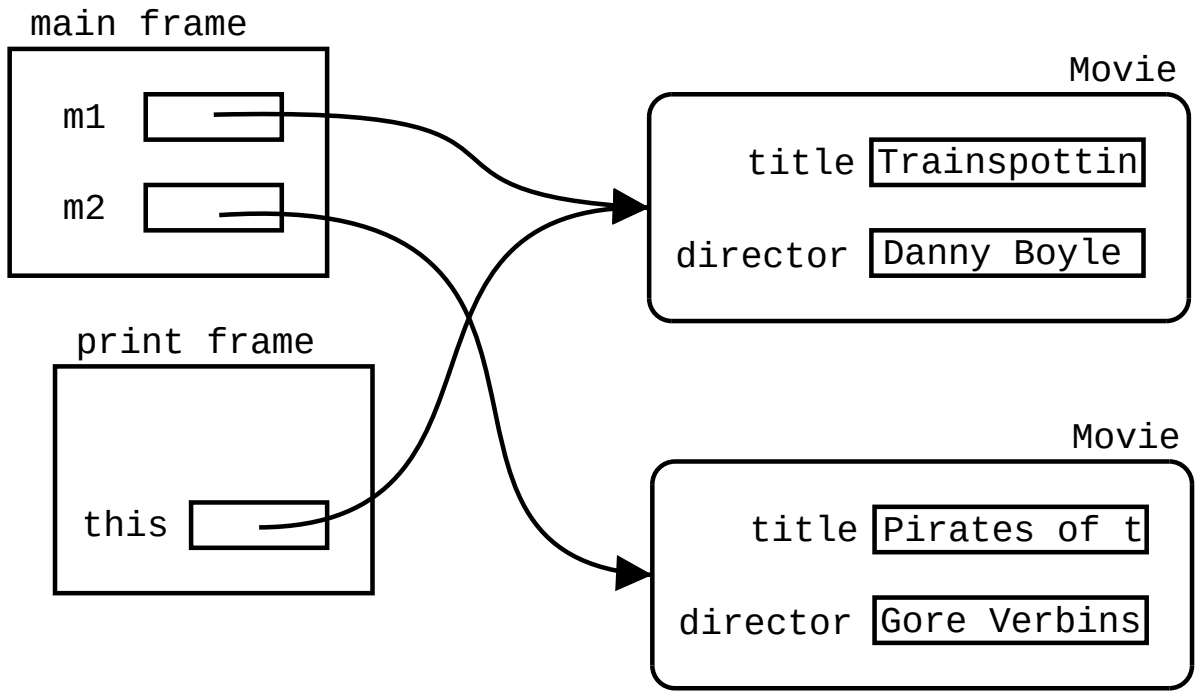
Execution


main frame
m1
m2
print frame
this
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

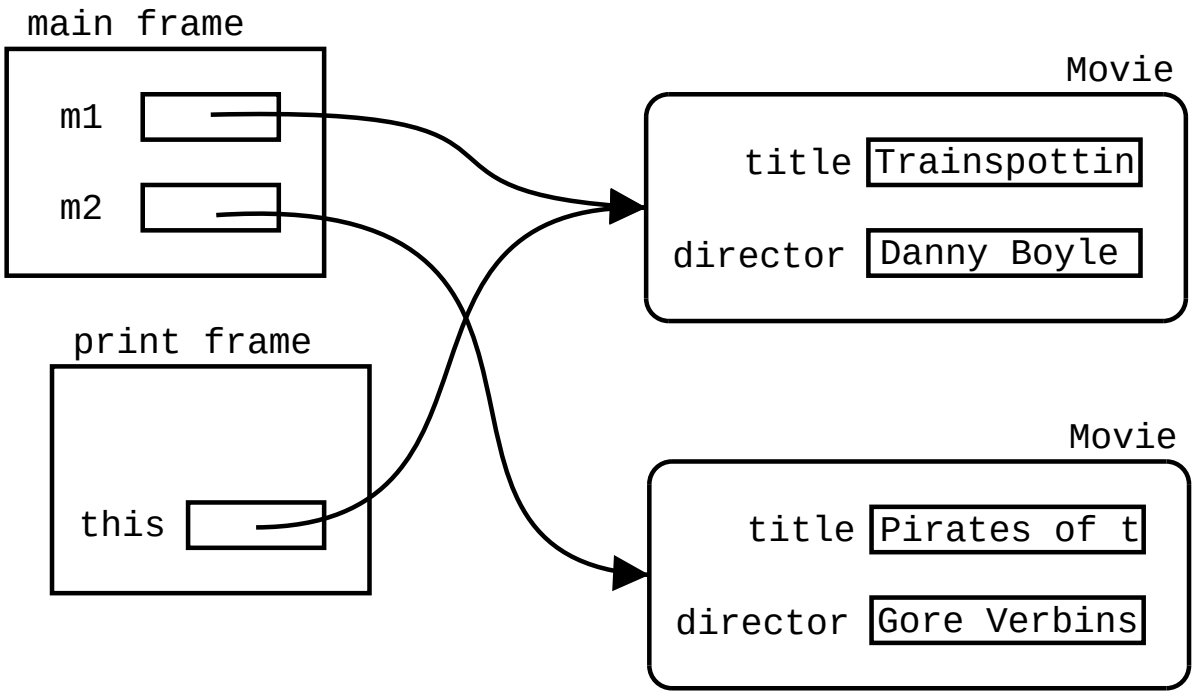
Execution


main frame
m1
m2
print frame
this
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

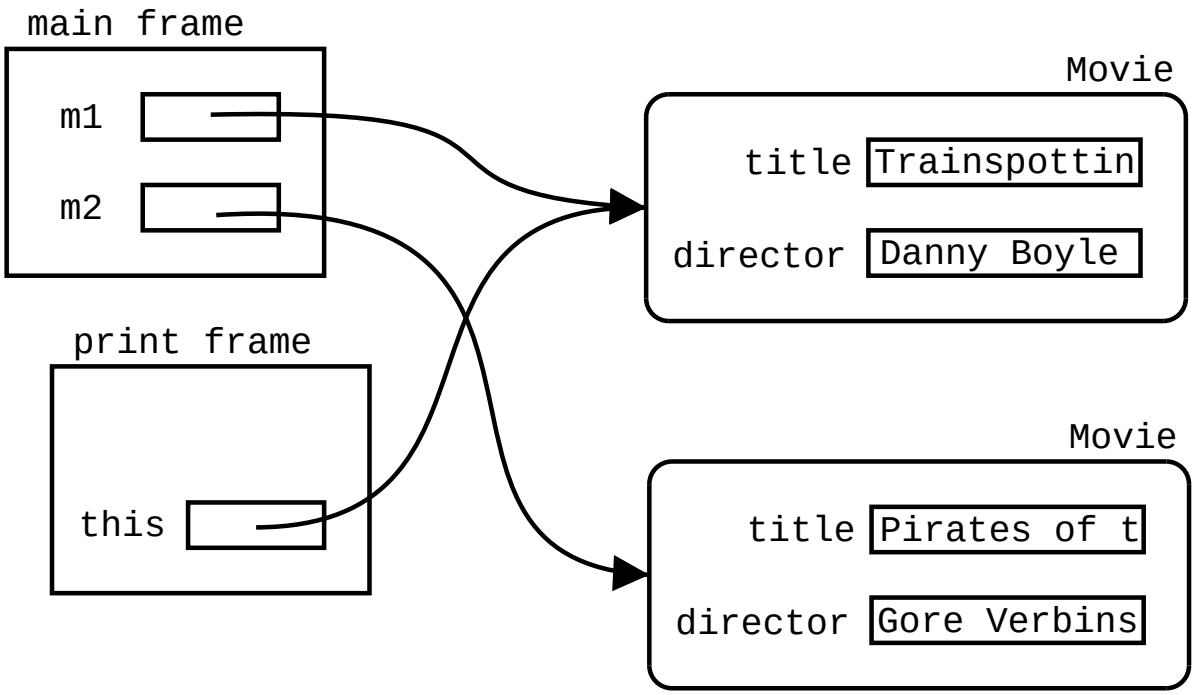
Execution


main frame
m1
m2
print frame
this
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

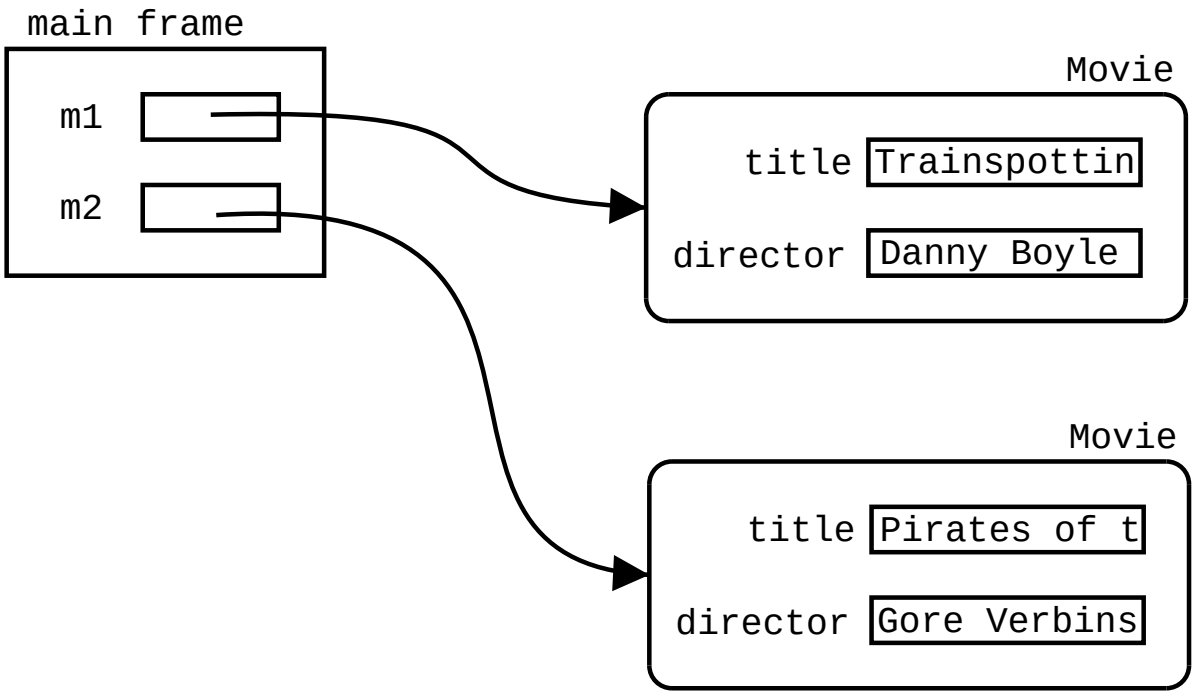
Execution


main frame
m1
m2
print frame
this
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

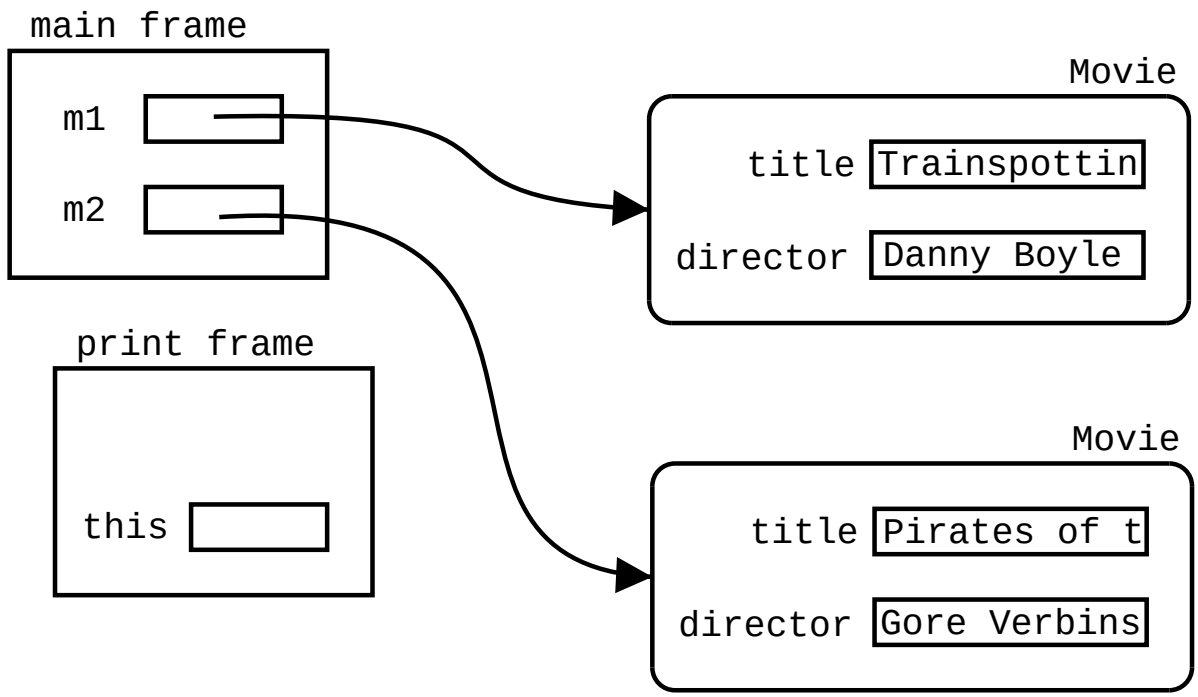
Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class MovieApplication {
  public static void main(String[] args)
  {
    Movie m1, m2;
    m1 = new Movie(“Trainspotting”, “Danny Boyle”)
    m2 = new Movie(“Pirates of the Caribbean”,
                   “Gore Verbinski”);
    m1.print();
    m2.print();
  }
}
```

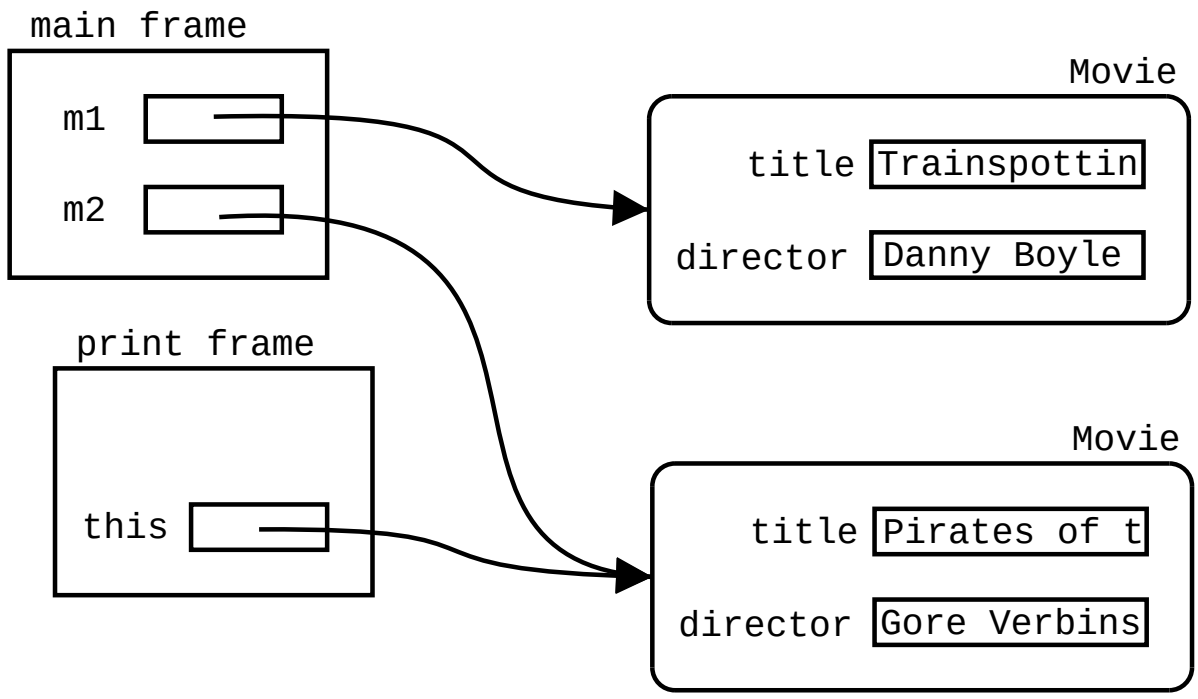
Execution


main frame
m1
m2
print frame
this
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

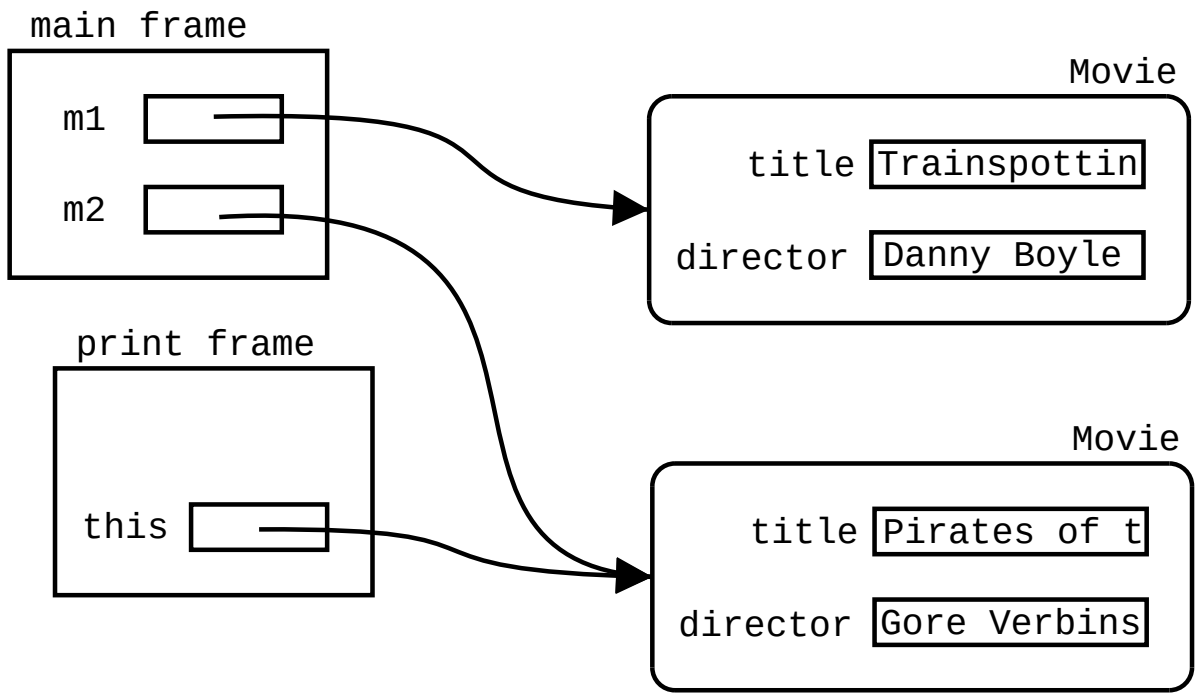
Execution


main frame
m1
m2
print frame
this
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

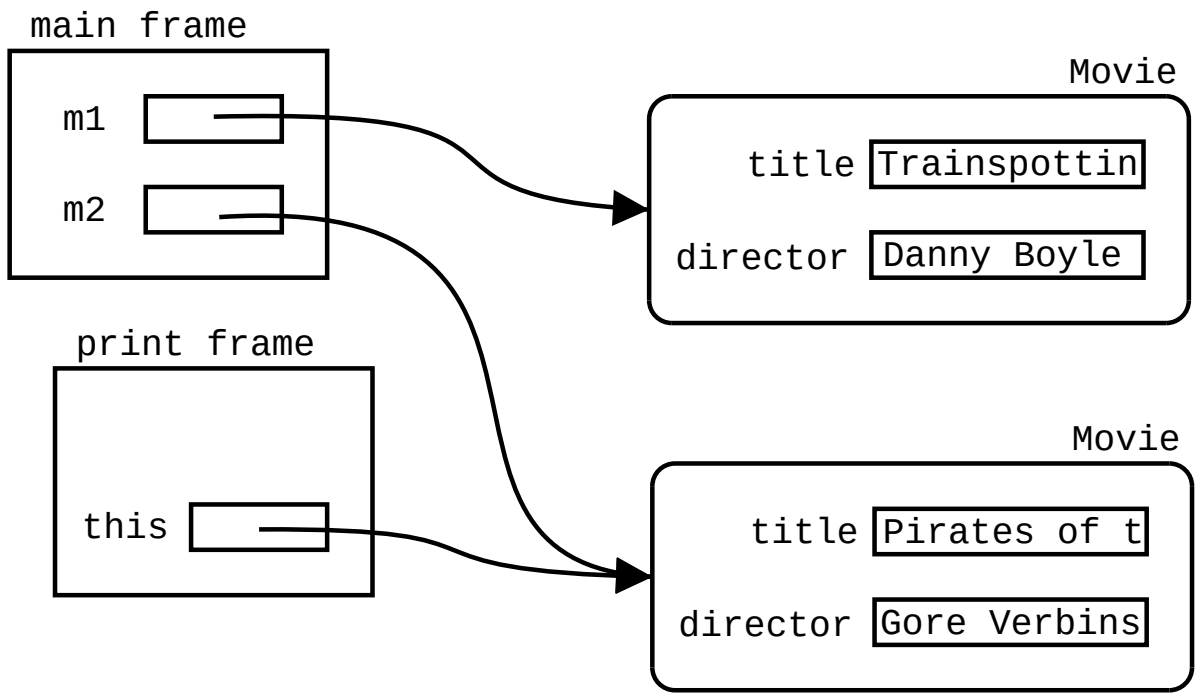
Execution


main frame
m1
m2
print frame
this
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

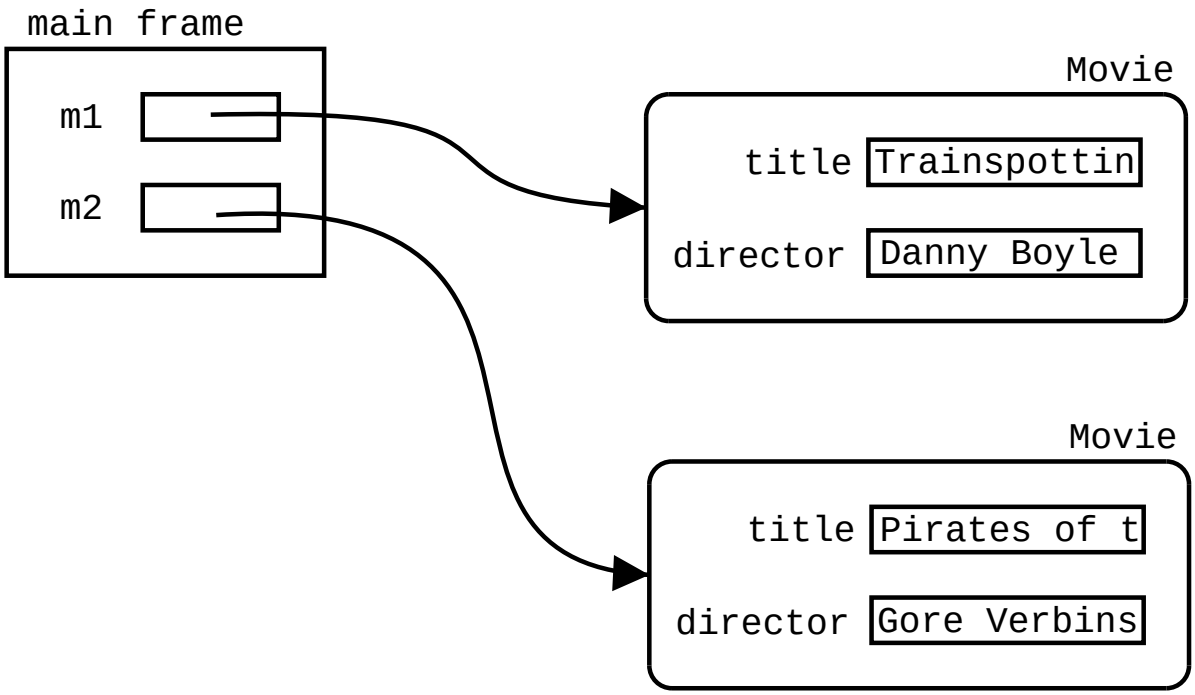
Execution


main frame
m1
m2
print frame
this
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

Execution

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    this.title = t;
    this.director = d;
  }
  void print()
  {
    System.out.println(this.title);
    System.out.println(this.director);
  }
}
```

Execution


main frame
m1
m2
Movie
title
Trainspottin
director
Danny Boyle
Movie
title
Pirates of t
director
Gore Verbins

When would it be necessary to use the "this" reference?

Whenever there is ambiguity: a local variable or parameter has the same identifier as an attribute

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void change_title(String t)
  {
    title = t;
  }
}
```

When to use "this"

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void change_title(String title)
  {
    title = title;
  }
}
```

When to use "this"

```
public class Movie
{
  String title, director;
  Movie(String t, String d)
  {
    title = t;
    director = d;
  }
  void change_title(String title)
  {
    this.title = title;
  }
}
```

Scope

- Names in different scopes do not conflict (like cities in different countries may have the same names)
- A variable can be used only if it is reachable, i.e. only if it is defined the current scope:
  - It must be in the current frame (it is a parameter, or a local variable),
  - ...or it must be defined in the class where it is being used (in this case it refers to an attribute of the object which received the message)

Scope

- Inside a method we can use only the following variables:
  - Parameters of the method
  - Local variables (declared inside the method)
  - Attributes of the method's class

Scope

```
public class House {
  double height;
  double width;

  void open_door()
  {
    boolean door_opened = true;
    System.out.println(door_opened);
  }
  void print_dimensions()
  {
    System.out.println(height);
    System.out.println(width);
  }
  void remove(double x)
  {
    height = height - x;
  }
}
```

Scope

```
public class House {
  double height;
  double width;

  void open_door()
  {
    boolean door_opened = true;
    System.out.println(door_opened);
  }
  void print_dimensions()
  {
    System.out.println(height);
    System.out.println(door_opened); //Error!
  }
  void remove(double x)
  {
    height = height - x;
  }
}
```

Scope

```
public class House {
  double height;
  double width;

  void open_door()
  {
    boolean door_opened = true;
    System.out.println(door_opened);
  }
  void print_dimensions()
  {
    System.out.println(height);
    System.out.println(x);   // Error!
  }
  void remove(double x)
  {
    height = height - x;
  }
}
```

Scope

```
public class House {
  double height;
  double width;

  void open_door()
  {
    boolean door_opened = true;
    System.out.println(door_opened);
  }
  void print_dimensions()
  {
    System.out.println(height);
    System.out.println(width);
  }
  void remove(double x)
  {
    height = height - b; // Error
  }
}
```

Scope

```
public class Neighbourhood {
  House house1;

  void build()
  {
    house1 = new House();
    house1.open_door();
  }
}
```

Scope

```
public class House {
  double height;
  double width;

  void open_door()
  {
    boolean door_opened = true;
    System.out.println(door_opened);
    house1.print_dimensions(); // Error
  }
  void print_dimensions()
  {
    System.out.println(height);
    System.out.println(width);
  }
  void remove(double x)
  {
    height = height - x;
  }
}
```

Scope

```
public class House {
  double height;
  double width;

  void open_door()
  {
    boolean door_opened = true;
    System.out.println(door_opened);
    this.print_dimensions(); // OK
  }
  void print_dimensions()
  {
    System.out.println(height);
    System.out.println(width);
  }
  void remove(double x)
  {
    height = height - x;
  }
}
```

Objects are "first class citizens"

- Since classes are data types and objects are their values, then we can do with objects the "same" things that we can do with primitive values, namely:
  - We can assign objects to variables,
  - We can pass objects as arguments to methods, and
  - Methods can return objects as their result.

OOP and Simulation

- Objects stand for entities of a "real" system
- Behaviour is determined by message passing
- OOP is very useful to model complex situations

Problem

Simulate a small factory consisting of only two workers.

A factory like this works as follows: it receives a production order consisting of a product name, the number of items that need to be produced and the difficulty of producing each item of this kind ('E' for easy, 'N' for normal, and 'D' for difficult.) Then for each item, one of the two workers is selected at random, and is told to build that item. The worker is given the product name and the difficulty of producing it, and it returns the new product once it has been built. Once the worker finnishes its job on that product, the factory reports the product's name, the name of the worker who assembled it and the amount of time it took to build. When all the items have been produced, the factory reports how much time took to complete all products, and for each worker, how many jobs it did and how much time it spent working.

Requirements

- Each individual product has a name as specified by the user, but with a number added to it to distinguish it from other items of the same line.

- For a normal job, a worker requires 10 minutes on average to finnish it, with a margin of plus or minus 2.5 minutes (i.e. it will take the worker between 7.5 and 12.5 minutes, and this is a random margin.) An easy job is done by a worker in 5 minutes on average, with a margin of 2.5 minutes. Finally a difficult job takes on average 15 minutes with a margin of 2.5 minutes as well.

- A worker has a a name, a number of completed jobs, and a total time worked.

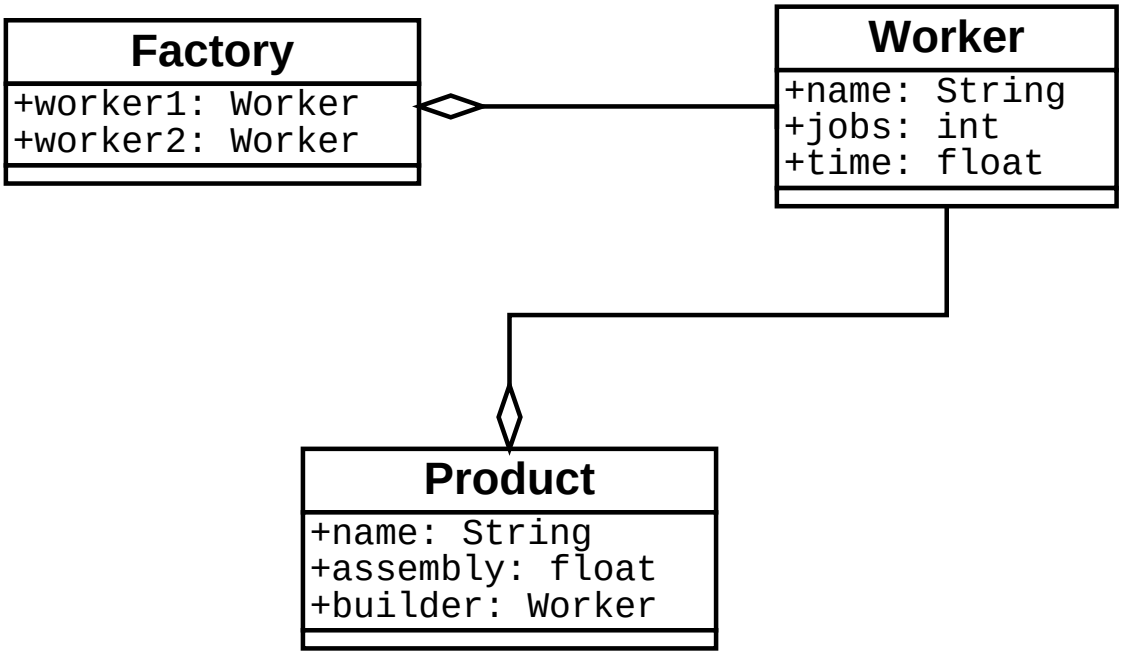
- A worker must be able to handle job requests from the factory. These requests consist of a name of the product and the difficulty. When a worker finnishes assembling a product, it returns the new product.

- A product has a name, the time it took to build it, and a reference to the worker who built it.

Analysis

- Relevant information:
  - Factories
  - Workers
  - Products
- Relationships (specially the "has-a" relationship:)
  - A factory has two workers.
  - A worker has a a name, a number of completed jobs, and a total time worked.
  - A product has a name, the time it took to build it, and a reference to the worker who built it.
- Activities and behaviours:
  - Workers can produce products
  - Factories can tell workers to work on a product of a particular kind and difficulty

Design


Factory
+worker1: Worker
+worker2: Worker
Worker
+name: String
+jobs: int
+time: float
Product
+name: String
+assembly: float
+builder: Worker

Implementation

```
public class Product {
  // ...
}
public class Worker {
  // ...
}
public class Factory {
  // ...
}
```

Implementation

```
public class Product {
  String name;
  float  assembly_time;
  Worker builder;
}
```

Implementation

```
public class Worker {
  String name;
  int    number_of_completed_jobs;
  float  total_time_worked;
}
```

Implementation

```
public class Factory {
  Worker worker1, worker2;
}
```

Implementation

```
public class Product {
  String name;
  float  assembly_time;
  Worker builder;
  Product(String name, float task_duration,
          Worker worker)
  {
    this.name = name;
    this.assembly_time = task_duration;
    this.builder = worker;
  }
}
```

Implementation

```
public class Product {
  String name;
  float  assembly_time;
  Worker builder;
  Product(String n, float t, Worker w) {
    this.name = n;
    this.assembly_time = t;
    this.builder = w;
  }
  String get_name()
  {
    return this.name;
  }
  float get_assembly_time()
  {
    return this.assembly_time;
  }
  Worker get_worker()
  {
    return this.worker;
  }
}
```

Implementation

```
public class Worker {
  String name;
  int    number_of_completed_jobs;
  float  total_time_worked;
  Worker(String name) {
    this.name = name;
    number_of_completed_jobs = 0;
    total_time_worked = 0.0f;
  }
  String get_name()
  {
    return name;
  }
  int number_of_completed_jobs()
  {
    return number_of_completed_jobs;
  }
  float total_time_worked()
  {
    return total_time_worked;
  }
}
```

Implementation

```
public class Worker {
  String name;
  int    number_of_completed_jobs;
  float  total_time_worked;
  Worker(String name) {
    this.name = name;
    number_of_completed_jobs = 0;
    total_time_worked = 0.0f;
  }
  String get_name()
  {
    return name;
  }
  int number_of_completed_jobs()
  {
    return number_of_completed_jobs;
  }
  float total_time_worked()
  {
    return total_time_worked;
  }
  // Continues below
```

```
    Product work(String product_name,
                 char difficulty)
    {
      // ...
    }
  } // End of class Worker
```

Implementation

```
Product work(String product_name, char difficulty)
{
  float average_task_duration = 10.0f;
  float margin = 2.5f;
  float task_duration = 0.0f;

  if (difficulty == 'E') {
    average_task_duration = average_task_duration - 5.0f;
  }
  else if (difficulty == 'D') {
    average_task_duration = average_task_duration + 5.0f;
  }

  // Continues...
```

```
    task_duration = average_task_duration
         + (float)((Math.random() * 2 - 1) * margin);

    number_of_completed_jobs++;
    total_time_worked = total_time_worked + task_duration;

    return new Product(product_name, task_duration, this);

  } // End of work method
```

Implementation

```
public class Worker
{
  // ...
  Worker(String name) { ... }
  String   get_name() { ... }
  int      number_of_completed_jobs() { ... }
  float    total_time_worked() { ... }
  Product  work(String product_name,
                char difficulty) { ... }
} // End of class Worker
```

Implementation

```
public class Factory
{
  Worker worker1, worker2;

  Factory(Worker worker1, Worker worker2) { ... }

  float produce(String product_name, int number_of_items,
                char difficulty) { ... }

  Worker get_worker1() { ... }
  Worker get_worker2() { ... }
}
```

Implementation

```
public class Factory
{
  Worker worker1, worker2;
  Factory(Worker worker1, Worker worker2)
  {
    this.worker1 = worker1;
    this.worker2 = worker2;
  }
  float produce(String product_name, int number_of_items,
                char difficulty)
  {
    int item = 1;
    float assembly_time = 0.0f;
    float total_time = 0.0f;
```

```
    while (item <= number_of_items) {
      int worker = choose_worker();
      Product p;
      if (worker == 1) {
        p = worker1.work(product_name + item, difficulty);
      }
      else {
        p = worker2.work(product_name + item, difficulty);
      }
      assembly_time = p.get_assembly_time();
      total_time = total_time + assembly_time;
      System.out.println("Product: "+p.get_name()+" was built by: "+
      item++;
    }
    return total_time;
  }
```

```
    Worker get_worker1()
    {
      return this.worker1;
    }
    Worker get_worker2()
    {
      return this.worker2;
    }
    int choose_worker()
    {
      return (int)(Math.random() * 2) + 1;
    }
  }
```

The end