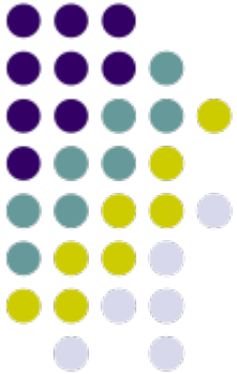
Singleton

CS 124

Reference: Gamma et al ("Gang-of-4"), *Design Patterns*

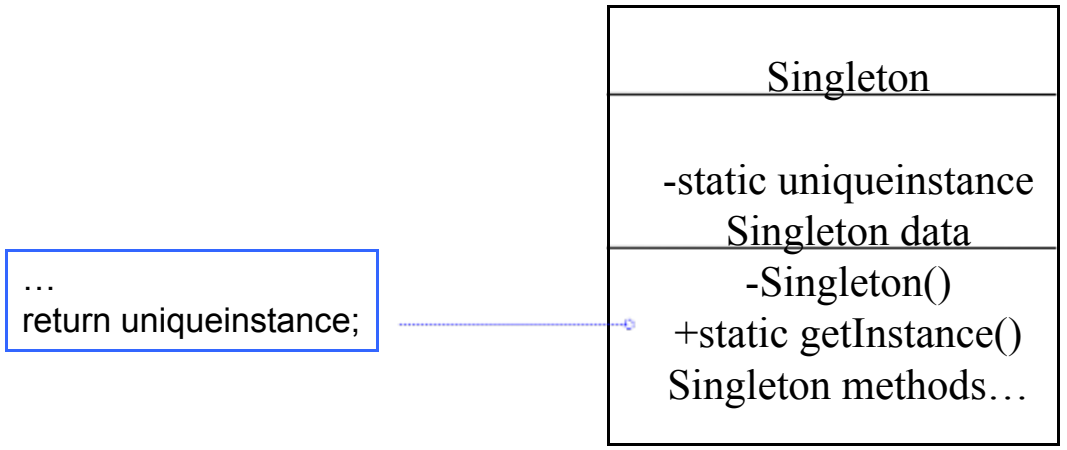


Singleton

- Intent
  - Ensure a class has only one instance, and provide a global point of access to it
- Motivation
  - Important for some classes to have exactly one instance. E.g., although there are many printers, should just have one print spooler
  - Ensure only one instance available and easily accessible
    - global variables gives access, but doesn't keep you from instantiating many objects
  - Give class responsibility for keeping track of its sole instance

Design Solution

- Defines a getInstance() operation that lets clients access its unique instance
- May be responsible for creating its own unique instance



Singleton Example (Java)

- Database

Database
static Database* DB instance attributes…
static Database* getDB() instance methods…

```
public class Database {
private static Database DB;
  ...
private Database() { ... }
public static Database getDB() {
  if (DB == null)
    DB = new Database();
  return DB;
}
  ...
}
```

In application code…

```
Database db =
Database.getDB();
db.someMethod();
```

Singleton Example (C++)

```
class Database
{
private:
  static Database *DB;
  ...
  private Database() { ... }
public:
  static Database *getDB()
  { if (DB == NULL)
      DB = new Database());
    return DB;
  }
  ...
}
Database *Database::DB=NULL;
```

In application code…

```
Database *db =
     Database.getDB();
Db->someMethod();
```

Implementation

- Declare all of class's constructors private
  - prevent other classes from directly creating an instance of this class
- Hide the operation that creates the instance behind a class operation (getInstance)
- Variation: Since creation policy is encapsulated in getInstance, it is possible to vary the creation policy

Singleton Consequences

- Ensures only one (e.g., Database) instance exists in the system
- Can maintain a pointer (need to create object on first get call) or an actual object
- Can also use this pattern to control fixed multiple instances
- Much better than the alternative: global variables