CS 211 : Thurs 02/15 (lecture 13)

Prof. Hummel
(he/him)

- **<u>Topics</u>: in-class programming (part 02), OOP with movies, and avoiding copies**

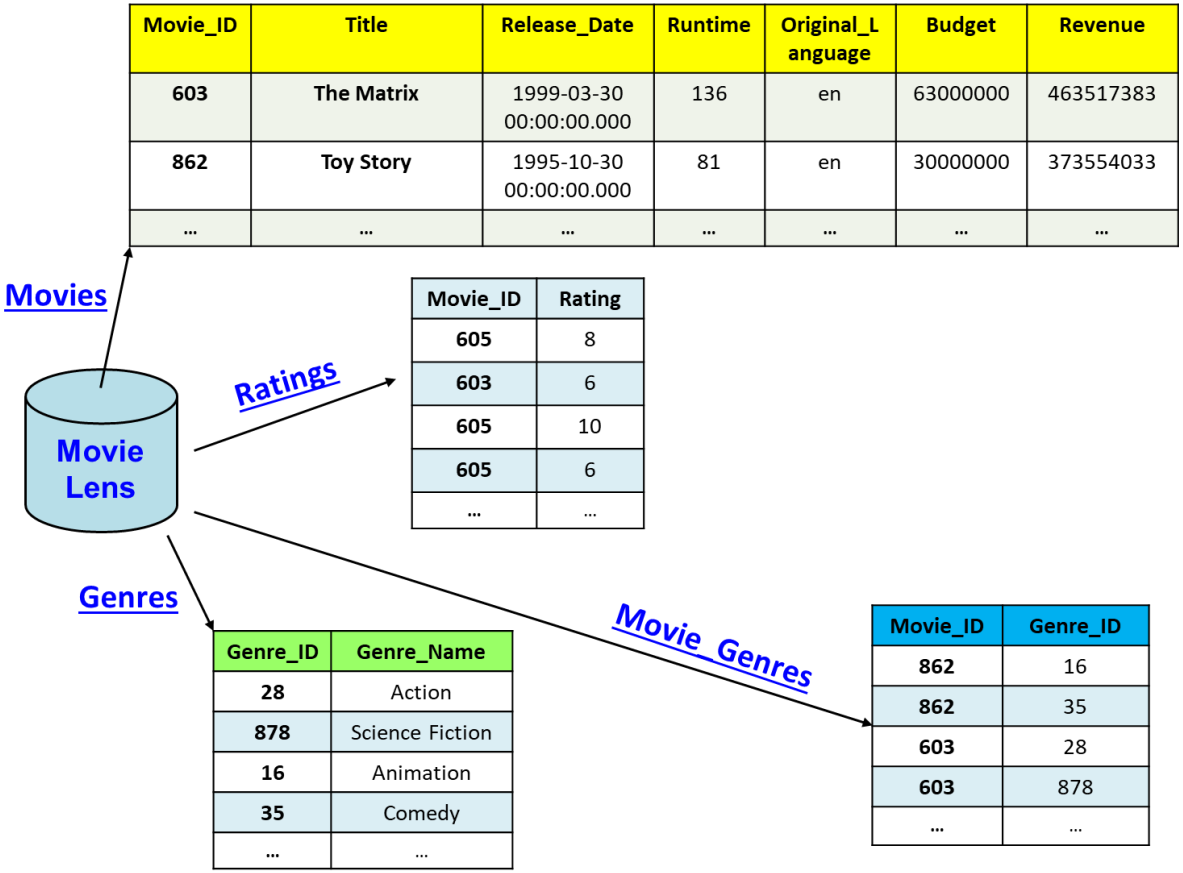
Notes:

- *Lecture slides available on Canvas*
- *We are going to program in class today, and will* ***collect at the end of class*** *via Gradescope*
- ***Project 05*** *due Friday night (can submit as late as Sunday with late days) – Gradescope open*
  - *Note that we are back on the EECS computers, replit not available*

Northwestern University

MovieLens database

- **MovieLens**
  - *https://movielens.org/*
  - *45K movies*



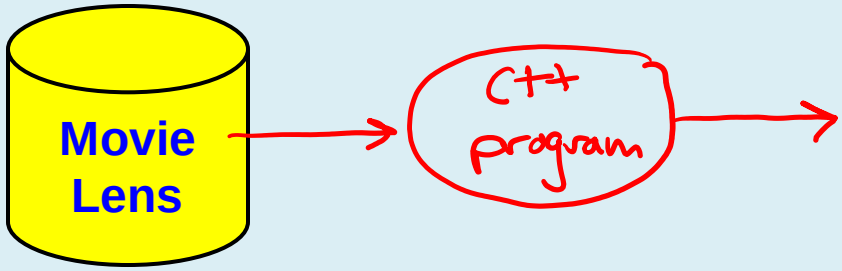
Movie_ID	Title	Release_Date	Runtime	Original_Language	Budget	Revenue
603	The Matrix	1999-03-30 00:00:00.000	136	en	63000000	463517383
862	Toy Story	1995-10-30 00:00:00.000	81	en	30000000	373554033
…	…	…	…	…	…	…

Movie_ID	Rating
605	8
603	6
605	10
605	6
…	…

Genre_ID	Genre_Name
28	Action
878	Science Fiction
16	Animation
35	Comedy
…	…

Movie_ID	Genre_ID
862	16
862	35
603	28
603	878
…	…

C++ Programming Demo: MovieLens




```
** MovieLens **

# of movies: 45431

862: Toy Story ($373554033.00), 5.08
Genres:
  Animation
  Comedy
  Family
8844: Jumanji ($262797249.00), 4.82
Genres:
  Adventure
  Fantasy
  Family
15602: Grumpier Old Men ($0.00), 5.79
Genres:
  Comedy
  Romance
31357: Waiting to Exhale ($81452156.00), 7.00
Genres:
  Drama
  Comedy
  Romance
11862: Father of the Bride Part II ($76578911.00), 6.16
Genres:
  Comedy
949: Heat ($187436818.00), 6.11
Genres:
  Drama
  Action
  Thriller
  Crime
```

Replit

- Login to replit.com
- Open team…
- Open project "Lecture 13"

EECS Computers

```
mkdir movies2
cd movies2
cp -r /home/cs211/w2024/lecture13/* .
make
./a.out
```

(1) Movie::getGenres( )

```
** MovieLens **

# of movies: 45431

862: Toy Story ($373554033.00), 5.08
Genres:
  Animation
  Comedy
  Family
8844: Jumanji ($262797249.00), 4.82
Genres:
  Adventure
  Fantasy
  Family
15602: Grumpier Old Men ($0.00), 5.79
Genres:
  Comedy
  Romance
31357: Waiting to Exhale ($81452156.00), 7.00
Genres:
  Drama
  Comedy
  Romance
11862: Father of the Bride Part II ($76578911.00), 6.16
Genres:
  Comedy
949: Heat ($187436818.00), 6.11
Genres:
  Drama
  Action
  Thriller
  Crime
```

- **We want to output a movie's genres…**
- **Define getGenres( ) in "movie.h" to return vector<string>**
- **Implement the function in "movie.cpp"**
  1. *Open database using private DB_*
  2. *Define SQL query to select Genre_Name using a join…*
  3. *Execute query*
  4. *Loop through the results and push each Genre_Name into vector*
  5. *Return vector*
  6. *Back in "main.cpp", after each movie call to get genres and output*
  7. *Run and test*

(2) Movie::print( )

- **Define void print( ) in "movie.h"**
- **Implement the function in "movie.cpp"**
  - *Move the code that outputs from main( ) to print( )*

- **Rewrite main( ) to call print( )…**

(3) Loop to interact with user

- **Back in main()**
- **Prompt and input a movie title using cin >> title**
- **Search vector for an <u>exact</u> match**
  - *Use a foreach loop and ==*
- **Repeat until user enters "#"**
- **Run and test…**
  1. *Notice if you enter a multi-word title, only inputs first word*
  2. *Switch to getline(cin, title)*

```
** MovieLens **

# of movies: 45431

Enter a movie title> Footloose
1788: Footloose ($80035402.00), 5.60
Genres:
  Drama
  Music
  Romance
  Family

Enter a movie title> Heat
949: Heat ($187436818.00), 6.11
Genres:
  Drama
  Action
  Thriller
  Crime

Enter a movie title> Fred
Sorry, movie not found...

Enter a movie title>
```

```
for(num2 = 0; num2<=9; num2++)
{
    for(num1=0;
    {
        cout<< nu
    }
}
```

outer

inner

To encourage the writing of helper functions / class methods, inner loops will not be allowed in future projects.

The inner loop has to be moved to a function…

(4) findMovie( ) function

- **Working in "main.cpp"**
- **Define void findMovie( ) function to do search**
  - *Pass movies and title as parameters*
  - *Move the search loop from main( ) to the function*

- **Now call the function from the loop in main( )…**

(5) Revised findMovie( ) function

- **Change the findMovie( ) function to find all movies that <u>contain</u> the given title string…**
  - *use string class* ***.find( )*** *function*
  - *.find(s) returns* ***string::npos*** *if string does not contain s*

```
** MovieLens **

# of movies: 45431

Enter a movie title> Matrix
603: The Matrix ($463317383.00), 6.00
Genres:
  Action
  Science Fiction
604: The Matrix Reloaded ($738599701.00), 6.90
Genres:
  Adventure
  Action
  Thriller
  Science Fiction
605: The Matrix Revolutions ($424988211.00), 5.42
Genres:
  Adventure
  Action
  Thriller
  Science Fiction
174615: Return to Source: The Philosophy of The Matrix ($0.00), 4.15
Genres:
  Documentary
21769: Armitage: Dual Matrix ($0.00), 5.37
Genres:
  Adventure
  Animation
  Action
  Thriller
  Science Fiction
14543: The Matrix Revisited ($0.00), 5.19
Genres:
  Documentary

Enter a movie title>
```

Submit your work to Gradescope

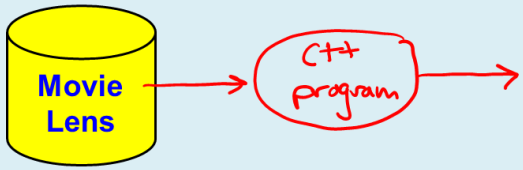
- **EECS computers?**

  make submit

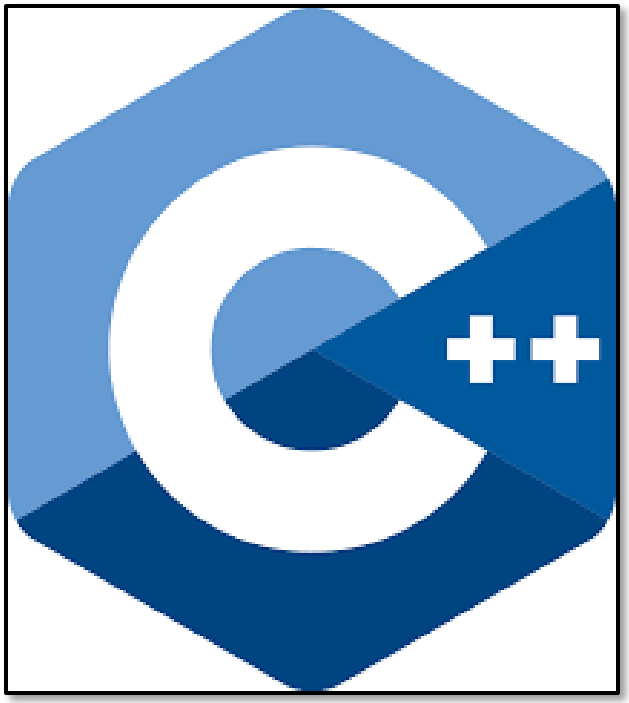
- **Replit?**

  – *download main.cpp, movie.cpp, movie.h*

  – *upload to Gradescope*




```
** MovieLens **

# of movies: 836

949: Heat ($187436818), 7.81
Genres:
  Drama
  Action
  Thriller
  Crime

710: GoldenEye ($352194034), 5.48
Genres:
  Adventure
  Action
  Thriller

1408: Cutthroat Island ($10017322), 7.41
Genres:
  Adventure
  Action

524: Casino ($116112375), 7.04
Genres:
  Drama
  Crime

5: Four Rooms ($4300000), 6.14
Genres:
  Comedy
  Crime
```



Observation

- **One of the biggest problems with C++ is objects being copied**
  - *Most copies are unnecessary and a waste of time*

- **How many extra copies of the Movie objects are being made?**
  - *Let's investigate…*

Add this code to main()

- **The Movie class tracks # of objects created & copied**
- **Add these output stmts to the end of main():**

```
int main()
{
  .
  .
  .

  cout << "** done **" << endl;

  cout << "# of movies created: " << created << endl;
  cout << "# of movies copied:  " << copied << endl;

  return 0;
}
```

```
Enter a movie title> #

** done **
# of movies created: 45431
# of movies copied:  656138
```

Copy constructor

- **Constructs an object by <u>copying</u> an existing object**

```
class Movie {
private:
  string DB_name;

public:
  int ID;
  string Title;
  double Revenue;

  // constructor
  Movie(string db_name,
        int id,
        string title,
        double revenue);

  // copy constructor
  Movie(const Movie& other);
```

```
//
// copy constructor
//
Movie::Movie(const Movie &other)
  : DB_name(other.DB_name),
    ID(other.ID),
    Title(other.Title),
    Revenue(other.Revenue)
{ }
```

Where are the copies being made?

```
vector<Movie> getMovies(string db_name)
{
  vector<Movie> movies;

  database db(db_name);

  string sql = "Select Movie_ID, Title, Revenue From Movies;";

  auto results = db << sql;

  for (auto row : results) {
    int id;
    string title;
    double revenue;

    row >> id >> title >> revenue;

    Movie m(db_name, id, title, revenue);
    movies.push_back(m);
  }

  return movies;
}
```

```
void findMovie(vector<Movie> movies, string title)
{
  //
  // search for an exact match:
  //
  bool found = false;

  for (Movie m : movies) {
    //if (m.Title == title) {
    if (m.Title.find(title) != string::npos) {
      //
      // movie title contains given string:
      //
      m.print();
      found = true;
      //break;
    }
  }//for

  if (!found)
    cout << "Sorry, movie not found..." << endl;
}
```

Solution: pass-by-ref

```
vector<Movie> getMovies(string db_name)
{
  vector<Movie> movies;

  database db(db_name);

  string sql = "Select Movie_ID, Title, Revenue From Movies;";

  auto results = db << sql;

  for (auto row : results) {
    int id;
    string title;
    double revenue;

    row >> id >> title >> revenue;

    Movie m(db_name, id, title, revenue);
    movies.push_back(m);
  }

  return movies;
}
```

emplace_back(db_name, id, title, revenue);

const

&

```
void findMovie(vector<Movie> movies, string title)
{
  //
  // search for an exact match:
  //
  bool found = false;

  for (Movie m : movies) {
    //if (m.Title == title) {
    if (m.Title.find(title) != string::npos) {
      //
      // movie title contains given string:
      //
      m.print();
      found = true;
      //break;
    }
  }//for

  if (!found)
    cout << "Sorry, movie not found..." << endl;
}
```

const

&

const

- **When you pass by const ref, C++ does not allow you to modify the objects**
  - *Any functions you call must be declared "const"*

```
class Movie {
private:
  string DB_name;

public:
  int ID;
  string Title;
  double Revenue;
  .
  .
  .

  // methods:
  double          getAverageRating() const;
  vector<string>  getGenres() const;
  void            print() const;
};
```

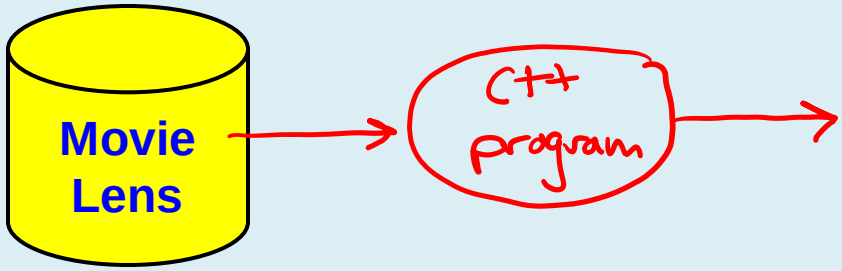
```
void findMovies(const vector<Movie>& movies, string title) {

  for (const Movie& m : movies) {
    //
    // see if m's title contains the given title string:
    //
    if (m.Title.find(title) != string::npos) {
      m.print();
    }
  }
}
```

"I declare these functions do not modify any of the data members…"

/**
* REFACTOR
* YOUR CODE.
*/

```
public static function bearerRequest($id, $secret)
{
    $uri    = self::$apiUri . "/oauth2/token";

    $body   = self::urlEncoded([
        "grant_type"    => "client_credentials",
        "client_id"     => $id,
        "client_secret" => $secret
    ], false);

    $response = self::doRequest($uri, $body, "post");

    if($response->code !== 200) throw new \Exception("API responde

    return $response->body;
}
```

What would you refactor to improve design?




```
** MovieLens **

# of movies: 45431

Enter a movie title> Matrix
603: The Matrix ($46[illegible]), 6.00
Genres:
  Action
  Science Fiction
604: The Matrix Reloaded ($738599701.00), 6.90
Genres:
  Adventure
  Action
  Thriller
  Science Fiction
605: The Matrix Revolutions ($424988211.00), 5.42
Genres:
  Adventure
  Action
  Thriller
  Science Fiction
174615: Return to Source: The Philosophy of The Matrix ($0.00), 4.15
Genres:
  Documentary
21769: Armitage: Dual Matrix ($0.00), 5.37
Genres:
  Adventure
  Animation
  Action
  Thriller
  Science Fiction
14543: The Matrix Revisited ($0.00), 5.19
Genres:
  Documentary

Enter a movie title>
```

Movies class

- **Create a class to model the collection of movies**
  - *vector<Movie>*
  - *getMovies( ) => constructor*
  - *findMovie( )*

(1) declare Movies class

- **Create "movies.h"**
- **Data members:** vector<Movie> Collection, string DB_name
- **Declare constructor to read movies from the database**
- **Declare findMovie(title) to return vector<Movie>**
  - *Do not output with cout, instead push movies into a vector & return*
  - *[As a general rule, classes never output to the screen]*

(2) implement Movies class

- Create "movies.cpp"
- Define constructor
- Define findMovies(title)

(3) rewrite main()

- **Rewrite main() to use Movies class…**
- **Run and test --- program should behave as before**

What's due?

Project 05 *due Friday night… Note that we are back on the EECS computers, replit not available*