THE BIG FOUR
FRIEND FUNCTIONS

Problem Solving with Computers-II

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

Read the syllabus. Know what's required. Know how to get help.

CLICKERS OUT

How is h01 (specifically the CS16 final) going?

A. Done - I think I have done well

B. Attempted - found it a bit difficult

C. Attempted - found some concepts alien

D. Attempted - extremely difficult

E. Haven’t attempted

Clickers out – frequency AB

The Big Four

1. Constructor
2. Destructor
3. Copy constructor
4. Copy Assignment

Constructor and Destructor

Every class has the following special methods:

- Constructor: Called right AFTER new objects are created in memory

- Destructor: Called right BEFORE an object is deleted from memory

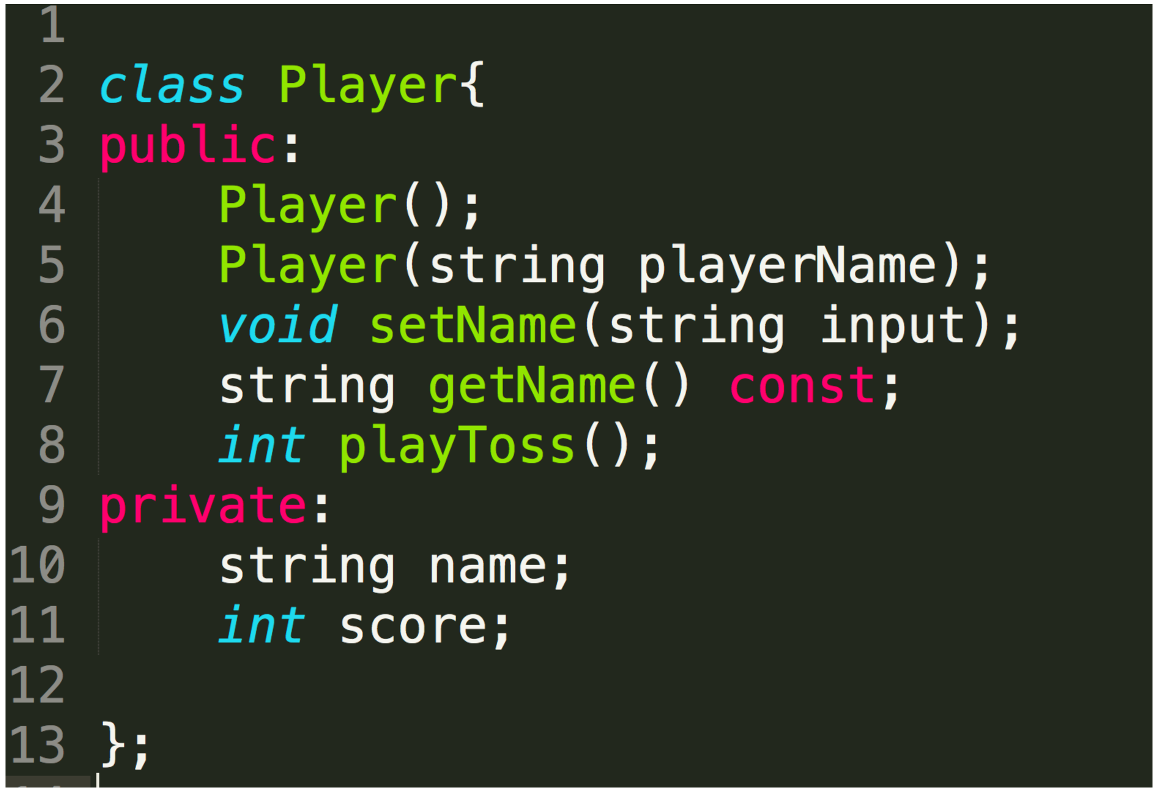
The compiler automatically generates default versions, if no constructor is implemented.

Constructor (last class)

```
void foo(){
    Player p;
    Player* q = new Player;
    Player w("Jill");
}
```

How many times is the constructor invoked for the above code?

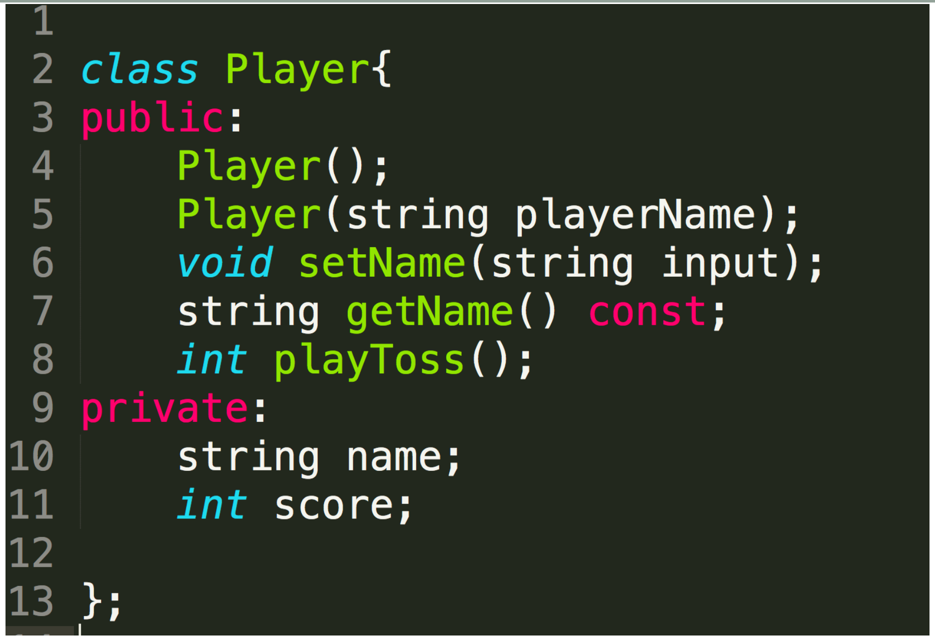
A. Never
B. Once
C. Twice
D. Thrice



```
class Player{
public:
    Player();
    Player(string playerName);
    void setName(string input);
    string getName() const;
    int playToss();
private:
    string name;
    int score;

};
```

Initialization lists

* Used to initialize member variables at the time they are created
* Must be used to initialize constant member variables



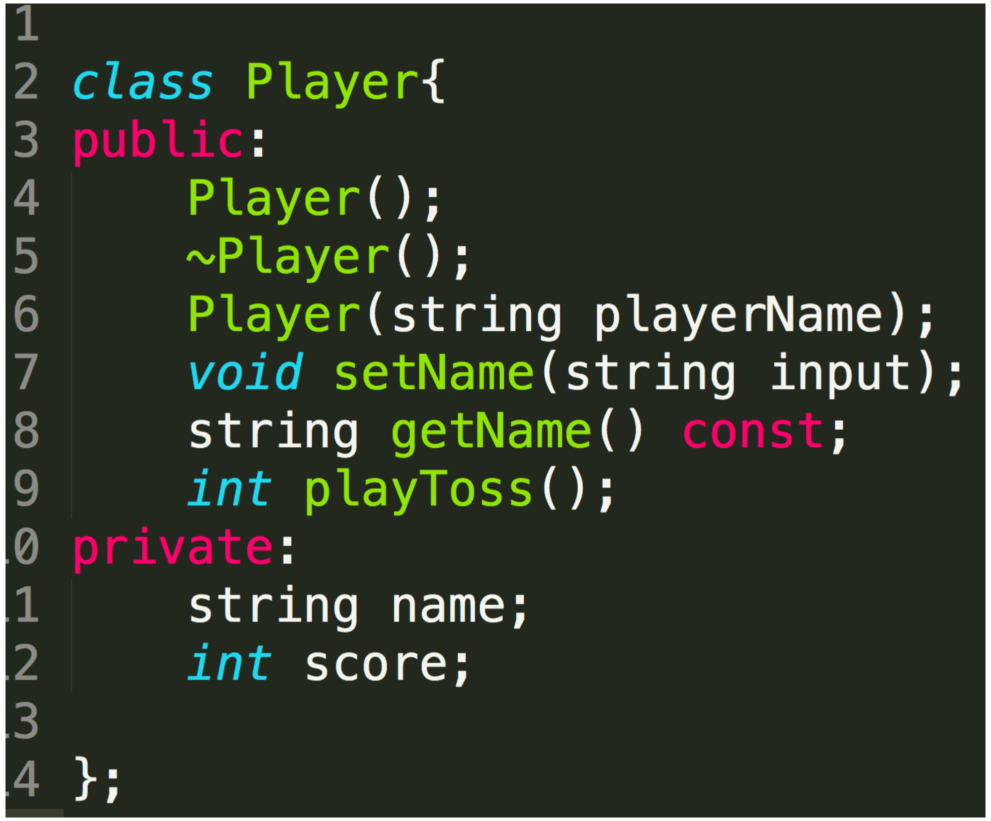
```

class Player{
public:
    Player();
    Player(string playerName);
    void setName(string input);
    string getName() const;
    int playToss();
private:
    string name;
    int score;

};
```

* For example, if the member variable “name” were a const, the constructor should use an initialization list as shown below:

Player::Player(string playerName):name(playerName), score(0) {
}

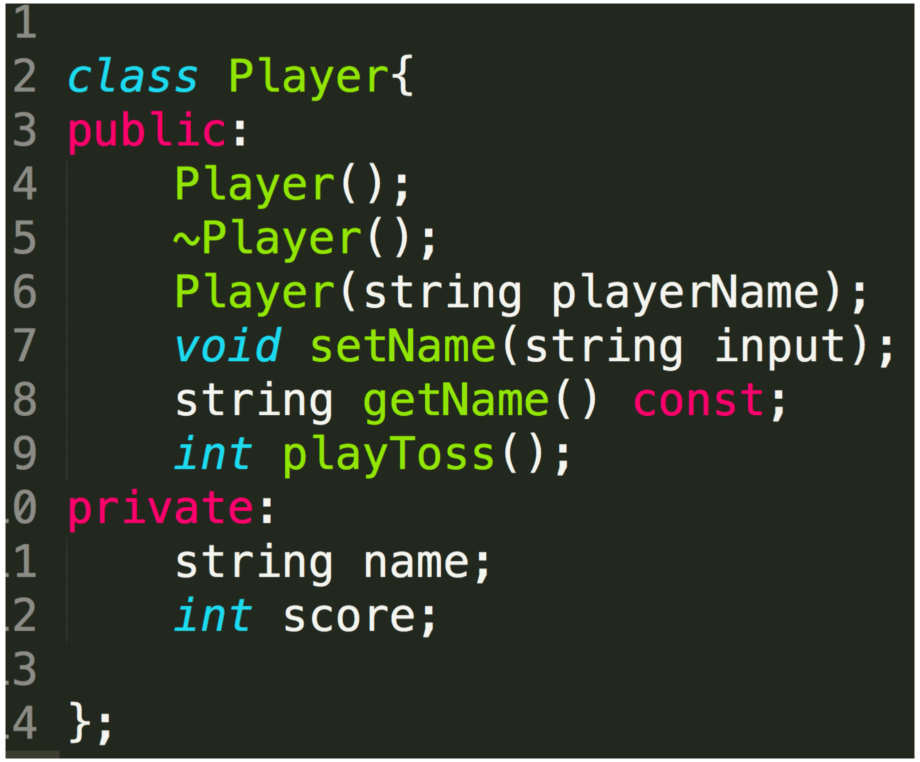
Destructor

- Must have the same name as the class preceded by a ~ (tilda)
- No return type
- Called right BEFORE an object is deleted from memory



```

class Player{
public:
    Player();
    ~Player();
    Player(string playerName);
    void setName(string input);
    string getName() const;
    int playToss();
private:
    string name;
    int score;

};
```

Destructor



```
class Player{
public:
    Player();
    ~Player();
    Player(string playerName);
    void setName(string input);
    string getName() const;
    int playToss();
private:
    string name;
    int score;

};
```

The destructor of which of the objects is called after foo() returns

A. p

B. q

C. *q

D. None of the above

```
void foo(){
    Player p;
    Player *q = new Player;
}
```

Copy constructor

- The copy constructor creates and initializes a new object to be the copy of another object of the class
- C++ provides a default copy constructor if one is not defined in the definition of the class
- The copy constructor is called in all the following cases, assuming p1 is an existing object of Player:

```
Player p2(p1);
Player p2 = p1;
Player *p2 = new Player(p1);
```

Copy constructor

- In which of the following cases is the copy constructor called?

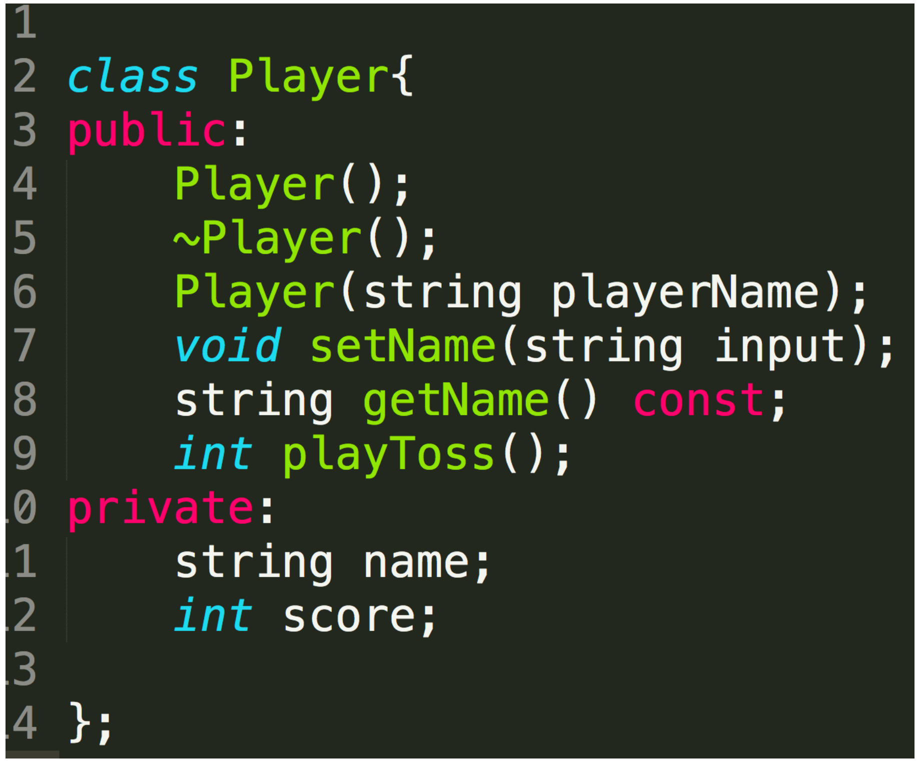
```
A. Player p1; Player p2(“Jill”);
B. Player p1(“Jill”); Player p2(p1);
C. Player *p1 = new Player(“Jill”); Player p2 = *p1;
D. B&C
E. A, B & C
```

Copy assignment

- Default behavior: Copies the member variables of one object into another

```
Player p1(“Jill”); // Parametrized constructor
Player p2;
p2 = p1; // Copy assignment function is called
```

Friend functions



```

class Player{
public:
    Player();
    ~Player();
    Player(string playerName);
    void setName(string input);
    string getName() const;
    int playToss();
private:
    string name;
    int score;

};
```

If a non-member function needs to access the PRIVATE members of a class, it should be declared as a friend function inside the class.

Example:
bool isEqual(Player& p1, Player& p2);

Returns True if p1 and p2 have the same name and score, otherwise false

Summary

- Classes have member variables and member functions (method). An object is a variable where the data type is a class.
- You should know how to declare a new class type, how to implement its member functions, how to use the class type.
- Frequently, the member functions of an class type place information in the member variables, or use information that's already in the member variables.
- New functionality may be added using non-member functions, friend functions, and operator overloading

Next time

- Operator Overloading