ECE PULSE

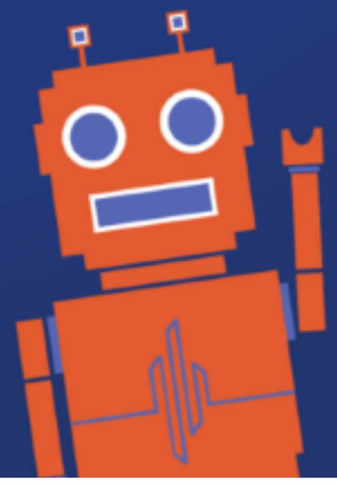
THE HEARTBEAT OF INNOVATION 2016

01.23.16 **Competitions**
01.28.16 to 01.30.16 **Main Conference**

Co-hosted By:

ViaSat

Register now at:
pulse.ece.illinois.edu/register



Today's announcements

- Course website: https://chara.cs.illinois.edu/cs225/
- General assistance (ews, svn, etc.) - post to piazza https://piazza.com/illinois/spring2016/cs225
- HW0 due 01/25 hard copy submission before lecture
- MP1 due 01/26, 11:59pm
- lab_intro due on 01/24, 11:59pm

- If you are not registered: if you want to get in the class, put your UIN on your HW0 submission.

First simple program and class definition

program.cpp

```
class sphere {
// member var and fn



};
```

- does it compile?
- does it run?

```
int main() {
    // lines to execute


    return 0;
}
```

Adding on to the simple program

program.cpp

```
class sphere {
//member var and fn




};
```

- does it compile?
- does it run?

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

int main() {

   cout << "Hello world!" << end;
   return 0;
}
```

Structure of a class defn (cont):

```
class sphere{
// member fn and data
public:




private:


};
```

`sphere` functionality:

1.

2.

3.

`sphere` representation:

```
int main(){




};
```

Equivalent ways of writing class declarations: Case1

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

//
class sphere {
public:
   double getDiameter();
   void setRadius(double newRad);
private:
   double radius;
};

//
void sphere::setRadius(double newRad){radius = newRad;}
double sphere::getDiameter() {return 2*radius;}

//
int main(){
   sphere a;
   a.setRadius(10.0);
   cout << "diameter is " << a.getDiameter() << endl;
}
```

Equivalent ways of writing class declarations: Case2

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

//
class sphere {
public:
   double getDiameter();
   void setRadius(double newRad);
private:
   double radius;
};

//
void sphere::setRadius(double newRad){radius = newRad;}
double sphere::getDiameter() {return 2*radius;}

//
int main(){
   sphere a;
   a.setRadius(10.0);
   cout << "diameter is " << a.getDiameter() << endl;
}
```

Equivalent ways of writing class declarations: Case1

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

//
class sphere {
public:
   double getDiameter();
   void setRadius(double newRad);
private:
   double radius;
};

//
void sphere::setRadius(double newRad){radius = newRad;}
double sphere::getDiameter() {return 2*radius;}

//
int main(){
   sphere a;
   a.setRadius(10.0);
   cout << "diameter is " << a.getDiameter() << endl;
}
```

Equivalent ways of writing class declarations: Case3

Encapsulation: .cpp and .h files

```
class sphere {
public:
  double getDiameter();
  void setRadius(double newRad);
private:
  double radius;
};
```

```
-:---  sphere.h        All L1      (C/l Abbrev)
#include "sphere.h"
void sphere::setRadius(double newRad){radius = newRad;}
double sphere::getDiameter() {return 2*radius;}
```

```
-:---  sphere.cpp      All L1      (C++/l Abbrev)
#include <iostream>
#include "sphere.h"
using namespace std;

int main(){
  sphere a;
  a.setRadius(10.0);
  cout << "diameter is " << a.getDiameter() << endl;
}
```

```
-:---  main.cpp        All L11     (C++/l Abbrev)
```

What is the result of compiling and running the following code?

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

//
class sphere {
public:
   double getDiameter();
   void setRadius(double newRad);
private:
   double radius;
};

//
void sphere::setRadius(double newRad){radius = newRad;}
double sphere::getDiameter() {return 2*radius;}

//
int main(){
   sphere a;
   a.setRadius(10.0);
   cout << "diameter is " << a.getDiameter() << endl;
}
```