Today's announcements

- MP3 is out, **E/C** is due on Tuesday, February 16 @11:59 PM
  MP3 is due on Friday, February 26 @11:59 PM

- **lab_gdb** release is due on Tuesday, February 16 at 11:59 PM

- Watch-at-home lecture on templates under Feb12 lecture:
  https://chara.cs.illinois.edu/cs225/lectures/
  or through direct link:
  https://recordings.engineering.illinois.edu:8443/ess/echo/presentation/0c0fe0e4-e1b1-4fc9-ba5f-06dc8c57642a

First in-lab exam this week!

- You **must** come to the lab section to which you are registered
- You **must** bring your i-card with you
- You **must** read Cinda's instructions posted on Piazza or sent through mass-mail
- **No notes, no cell phones, no water bottles, no food. Backpacks with all of the above must be stored in front of the lab.**
- You will get at least **1% extra credit** for the exam - as if you attended the lab!
- Exam materials: https://chara.cs.illinois.edu/cs225/exams/mt1/

What can and can't sphere/ball objects do?

```
class sphere {
public:
sphere();
sphere(double r);
double getVolume();
void setRadius(double r);
void display();
private:
double theRadius;};
```

```
class ball:public sphere {
public:
ball();
ball(double r string n);
string getName();
void setName(string n);
void display();
private:
string name;};
```

```
int main () {
  sphere a;
  ball b;
```

sphere() _____

sphere(double r) _____

getVolume() _____

setRadius(double r) _____

display() //sphere) _____

the Radius _____

ball() _____

ball(double r, string n) _____

getName() _____

setName(string n) _____

display() //ball _____

name _____

```
class sphere {
public:
  void display() {cout << "I am a Sphere!" << endl;}
  double getRadius() {return rad;}
  void setRadius(double r) {rad = r;}
private:
  int rad;
};

class ball: public sphere {
public:
  void display() {cout << "I am a Ball!!!" << endl;}
  string getName() {return name;}
  void setName(string n) {name = n;}
private:
  string name;
};
```

```
void myfun(sphere x){
  x.display();
}
```

```
int main () {
  sphere a;
  ball b;

  a = b;
  b = a;
```

```
int main () {
  sphere a;
  ball b;

  myfun(a);
  myfun(b);

  return 0;
}
```

something to consider:

```
class sphere {
public:
    sphere();
    sphere(double r);
    …
    void display();
private:
    double theRadius;
};
```

```
void sphere::display() {
    cout << "sphere" << endl;
}
```

```
class ball:public sphere {
public:
    ball();
    ball(double r string n);
    …
    void display();
private:
    string name;
};
```

```
void ball::display() {
    cout << "ball" << endl;
}
```

ex1

```
sphere s;
ball b;
s.display();
b.display();
```

ex2

```
sphere * sptr;
sptr = &s;
sptr->display();
```

ex3

```
sphere * sptr;
sptr = &b;
sptr->display();
```

“virtual” functions:

```
class sphere {
public:
    sphere();
    sphere(double r);
    …
                void display();
private:
    double theRadius;
};
```

```
void sphere::display() {
   cout << “sphere” << endl;
}
```

```
class ball:public sphere {
public:
    ball();
    ball(double r string n);
    string getName();
                void display();
private:
    string name;
};
```

```
void ball::display() {
   cout << “ball” << endl;
}
```

ex4

```
if (a==0)
   sptr = &s;
else sptr = &b;
sptr->display();
```

virtual functions – the rules:

A virtual method is one a _____ can override.

A class's virtual methods _____ be implemented. If not, then the class is an "abstract base class" and no objects of that type can be declared.

A derived class is not *required* to override an existing implementation of an _____ virtual method.

Constructors _____ be virtual

Destructors can and _____virtual

Virtual method return type _____ be overwritten.

Constructors for derived class:

```
ball::ball():sphere()
{
    name = “not known”;
}
```

```
ball b;
```

```
ball::ball(double r, string n):
sphere(r)
{
    name = n;
}
```

```
ball b(0.5,”grape”);
```

“virtual” destructors:

```
class Base{
public:
   Base(){cout<<”Ctor: B"<<endl;}
   ~Base(){cout<<”Dtor: B"<<endl;}
};

class Derived: public Base{
public:
   Derived(){cout<<”Ctor: D"<<endl;}
   ~Derived(){cout<<”Dtor: D"<<endl;}
};
```

```
void main(){
   Base * V = new Derived();
   delete V;
}
```

Abstract Base Classes:

```
class flower {
public:
    flower();
    virtual void drawBlossom() = 0;
    virtual void drawStem() = 0;
    virtual void drawFoliage() = 0;
    …
};
```

```
void daisy::drawBlossom() {
// whatever
}
void daisy::drawStem() {
// whatever
}
void daisy::drawFoliage() {
// whatever
}
```

```
class daisy:public flower {
public:
    virtual void drawBlossom();
    virtual void drawStem();
    virtual void drawFoliage();
    …
private:
    int blossom; // number of petals
    int stem; // length of stem
    int foliage // leaves per inch
};
```

```
flower f;
daisy d;
flower * fptr;
```

VIRTUAL REALITY

University of Illinois

Demo the new HTC Vive on campus!

Monday - Friday
February 15 - 19

NCSA Atrium
8 am - 5 pm

Sign ups begin at 8 am and
are first come, first-serve

Follow @HTCVive for
live updates

VIVE

htc | STEAM VR

@HTCVive HTCVive @HTCVive www.htcvr.com

Concluding remarks on inheritance:

Polymorphism: objects of different types can employ methods of the same name and parameterization.

```
animal ** farm;

farm = new animal*[5];
farm[0] = new dog;
farm[1] = new pig;
farm[2] = new horse;
farm[3] = new cow;
farm[4] = new duck;

for (int i=0; i<5;i++)
    farm[i]->speak();
```

Inheritance provides DYNAMIC polymorphism—type dependent functions can be selected at run-time. Wikipedia: Polymorphism in OOP

Next topic: “templates” are C++ implementation of static polymorphism, where type dependent functions are chosen at compile-time.

Exercise: What is the output produced by this code?

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

class sphere {
public:
  sphere() {cout << "create sphere" << endl;}
  double getRadius() {return rad;}
  void setRadius(double r) {rad = r;}
  virtual void display(){cout << "I am sphere!" << endl;}
private:
  double rad;
};


class ball: public sphere {
public:
  string getName() {return name;}
  void setName(string n){name = n;}
  virtual void display(){cout << "I am ball!" << endl;}
private:
  string name;
};

int main(){
  sphere s;
  ball b;

  sphere s1 = b;
  sphere *s2 = &b;

  s.display();
  b.display();
  s1.display();
  s2->display();
}
```