Using Objects

Last updated 3/10/19

Using Objects

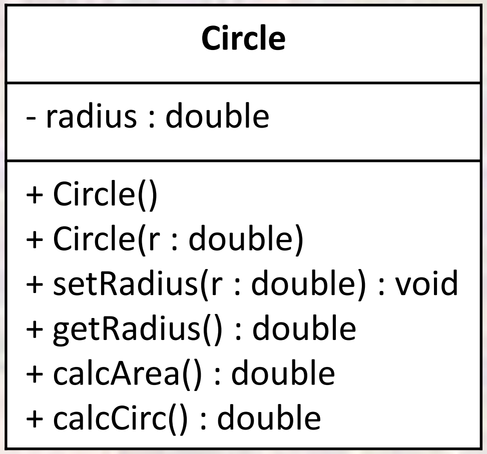
- Creating Objects

  - Create a circle with default radius

    Circle circle1;

  - Create a circle with initialized radius

    Circle circle2(3.2)



```
Circle
- radius : double
+ Circle()
+ Circle(r : double)
+ setRadius(r : double) : void
+ getRadius() : double
+ calcArea() : double
+ calcCirc() : double
```

Using Objects

- Changing Objects

  - Change the radius of circle1

    ```
    circle1.setRadius(5);
    ```

Circle
- radius : double
+ Circle() + Circle(r : double) + setRadius(r : double) : void + getRadius() : double + calcArea() : double + calcCirc() : double

Using Objects

- Reading/Using Objects
  - Read the radius of circle1

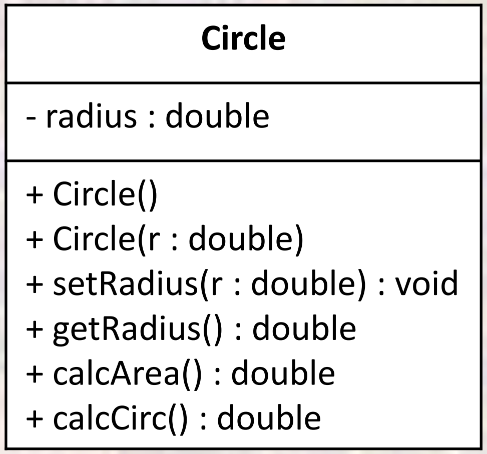
    ```
    foo = circle1.getRadius();
    ```

  - Calculate the area of circle1

    ```
    foo = circle1.calcArea();
    ```

  - Calculate the circumference of circle1

    ```
    foo = circle1.calcCirc();
    ```



Using Objects

- Passing objects to functions

  - Objects are treated just like any other variable

  - Passed to functions
    - By value
    - By reference
    - By pointer

Using Objects

Circle
- radius : double
+ Circle() + Circle(r : double) + setRadius(r : double) : void + getRadius() : double + calcArea() : double + calcCirc() : double

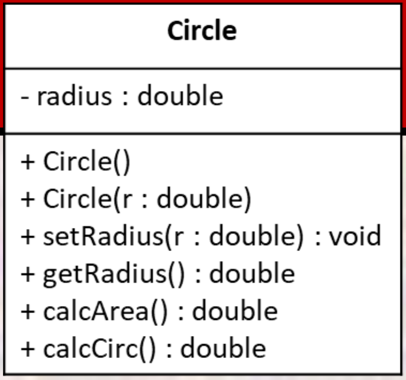
- Passing objects to functions by value
  - A copy of the object is passed to the function
  - Changes to the object are restricted to the functions scope
  - Function to print the characteristics of a circle

```
void print_circle(Circle cir);
```
Declaration

```
void print_circle(Circle cir){
   double rad;
   rad = cir.getRadius();
   cout << “Radius is: “ << rad << endl
   cout << “Area is: “ << cir.calcArea() << endl
} // end print_circle
```
Definition

```
print_circle(circle1);
```
Call

Using Objects



- Passing objects to functions by reference
  - A reference to the object is passed to the function
  - Changes to the object are not restricted to the functions scope
    - Similar to passing by pointer
  - Function to double the radius of a circle

```
void double_circle(Circle& cir);                                  Declaration

void double_circle(Circle& cir){                                  Definition
   double rad;
   rad = cir.getRadius();
   cir.setRadius(rad*2);
} // end double_circle

double_circle(circle1);      // double the radius of circle1      Call
```

Using Objects

- Passing objects to functions by value vs. reference
  - Passing by value
    - Every variable in the object to be copied (onto the stack)
    - The passed object cannot be changed (safe)
  - Passing by reference
    - No copy is made
    - The passed object can be changed (unsafe?)
  - 3rd Option – Pass by reference but with constant reference parameters
    - No copy is made
    - Variables cannot be changed
    - Only functions marked as constant can be used

Using Objects

- Passing objects to functions with constant reference parameters
  - Only functions marked as constant can be used

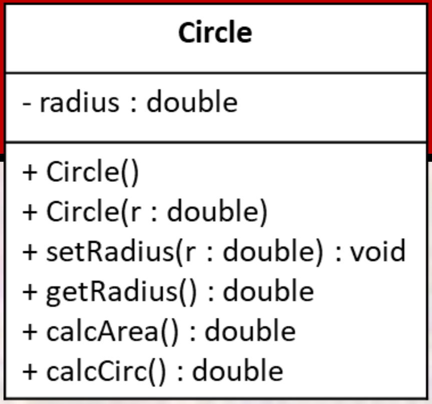
```
void print_circle(const Circle& cir);

void print_circle(const Circle& cir){
   double rad;
   rad = cir.getRadius();
   cout << "Radius is: " << rad << endl
   cout << "Cant print area – not a const fn" << endl
} // end print_circle

print_circle(circle1);
```

```
Circle.cpp
…
void Circle::setRadius(double r){
      radius = r;
}
double Circle::getRadius(void) const{
 return radius;
}
double Circle::calcArea(void){
   return (3.14 * radius * radius);
 }
```

Using Objects



```
Circle
- radius : double
+ Circle()
+ Circle(r : double)
+ setRadius(r : double) : void
+ getRadius() : double
+ calcArea() : double
+ calcCirc() : double
```

- Passing objects to functions by pointer
  - A pointer to the object is passed to the function
  - Changes to the object are not restricted to the functions scope
  - Function to double the radius of a circle

Declaration

```
void double_circle(Circle * cir);
```

Definition

```
void double_circle(Circle * cir){
  double rad;
  rad = (*cir).getRadius();       // * dereference
  cir->setRadius(rad*2);          // indirect dereference
} // end double_circle
```

Call

```
Circle circle1;
Circle * cir1_ptr = & circle1;
double_circle(& circle1);       // double the radius of circle1
double_circle(cir1_ptr);        // double the radius of circle1 again
```

Using Objects

- Returning objects from functions
  - Objects can be returned from functions just like any other variable

```
Circle double_circle(Circle  cir_ref);
```

Declaration

```
Circle double_circle(Circle  cir_ref){
   Circle cir_out;
   double rad;
   rad = cir_ref.getRadius();
   cir_out.setRadius(rad*2);
   return cir_out
} // end double_circle
```

Definition

Circle
- radius : double
+ Circle() + Circle(r : double) + setRadius(r : double) : void + getRadius() : double + calcArea() : double + calcCirc() : double

```
Circle circle1;
Circle circle2
circle2 = double_circle(circle1);
```

Call