

CISC 1600/1610 Computer Science I

Classes

Professor Daniel Leeds
dleeds@fordham.edu
JMH 328A

Data types

Single pieces of information

- one integer – int
- one symbol – char
- one truth value – bool

Multiple pieces of information

- group of symbols – string
- group of anything – **array**
- group of multiple things – struct, **class**

2

Introducing: classes

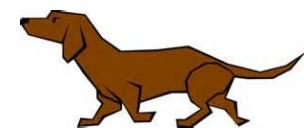
- A **class** defines a new data type
- Each instance of a class is an **object**
- Each object can contain
 - Actions to perform (functions)
 - Information about the object (variables)

3

Class dog

Example information

- Size
- Weight
- Location



Example actions

- Eat
- Walk
- Bark

4

Class syntax – declaration

```
class Dog
{
public:
    void Bark();
    void Eat(float foodQuantity);
    void Walk(float distance);
    float size, weight, location;
};
```

5

Class syntax – function definitions

```
void Dog::Bark()
{
    cout << "Woof woof!\n";
}

void Dog::Eat(float foodQuantity)
{
    weight+=foodQuantity/2;
    size+=foodQuantity/10;
}
```

6

Class syntax – create and use an object

```
int main()
{
    Dog fido;
    fido.weight=40.5;
    fido.size=10;
    fido.Eat(20);
    cout << fido.weight << " "
        << fido.size << endl;
    return 0;
}
```

7

Typical program layout

```
class Dog {
    . .
};

int main() {
    Dog fluffy;
    fluffy.Bark();
    . .
}

void Dog::Bark() {
    . .
}
```

Declaration**Usage****Definitions**

8

Class syntax – declaration

```
class Dog
{
public:
    float size, weight, location;
    void Bark();
    void Eat(float foodQuantity);
    void Walk(float distance);
};
```

Member variables**Member functions**

9

The Dot Operator .

- Used for functions and data of individual objects
- `fido.Bark()`

The Scope Resolution Operator ::

- Used for functions of a class
- `Dog::Bark()`

Note: a function inside a class is called a “member function”

10

Multiple instances of a class

```
int main()
{
    Dog fido, spot;
    fido.weight=40.5; fido.size=10;
    spot.weight=30; spot.size=7.5;
    fido.Eat(20); fido.Eat(2);
    cout << fido.weight << " "
        << spot.weight << endl;
    return 0;
}
```



11

Time to walk the dog...

```
int main()
{
    Dog rufus;
    rufus.weight=35; rufus.size=7.2;
    rufus.location=5;
    rufus.Walk(3.4);
    cout << "New location for Rufus: "
        << rufus.location << endl;
    // Will output location 8.4
    return 0;
}
```

12

Class syntax – create and use an object

```
int main()
{
    Dog fido;
    fido.weight=40.5;
    fido.Bark();
    ?? // Set fido's location to 3
    cout << ??; // output location
    ?? // Have fido change location
    return 0;
}
```

14

public vs. private

- **public:** any function can see and use
- **private:** only visible to member functions
- **Good style:**
 - make all member variables private
 - use public functions to access and mutate variables

15

Class declaration, take 2

```
class Cat
{
public:
    void set(float inWeight, float inSize,
              float inLoc);
    float getSize();
    float getWeight();
    float getLocation();
    ...
    void Walk(float distance);
private:
    float size, weight, location;
};
```

16

Function definitions – take 2

```
void Cat::set(float inWeight,
              float inSize,
              float inLoc)
{
    weight=inWeight;
    size=inSize;
    location=inLoc;
}
```

17

Class usage – take 2

```
int main()
{
    Cat feline1;
    feline1.set(5.5,20.1,2);
    cout << feline1.location; // Error
    cout << feline1.getLocation()
        << endl;
    return 0;
}
```

18

Bank account

Variables

- Name
- Current balance
- History of cash in (and out)

Functions

- Deposit:
 - Add entry to history
 - Update balance

19

```

class Account {
public:
    void open(string inName);
    void deposit(float money);
    float getBalance();

private:
    string name;
    float balance;
};

```

22

```

void Account::open(string inName) {
    name=inName;
    balance=0;
}

void Account::deposit(float money) {
    if(money>=0) {
        balance = balance+money;
    }
    else {
        cout << "Error! "
            << "Negative deposit!\n";
    }
}

```

23

What does this do?

```

int main()
{
    Account acc1;
    acc1.open("Tina");
    cout << acc1.getBalance() << endl;
    acc1.deposit(250);
    acc1.deposit(20.25);
    cout << acc1.getBalance() << endl;
}

```

24

Withdrawal function?

- How can we write withdraw function to reduce the money in our account?
- How can we prevent over-drawing?

25

Account review

Member variables

- name
- balance
- history,
num_transactions

Member functions

- | | |
|----------------|-----------------------|
| • open | New accessor function |
| • deposit | • string getName() |
| • getBalance | |
| • withdraw | |
| • printHistory | |

26