Previous Lecture:
- Two-dimensional array of objects
- Review String
- Instantiating 2-d arrays

Today’s Lecture:
- Inheritance

Reading:
- Sec 4.1, 4.4

Separate classes—each has its own members

```java
class Dice {
    private int top;
    private int sides;
    public Dice(...) {...}
    public void roll() {...}
    public String toString() {...}
    public int getTop() {...}
    public int getSides() {...}
}
class TrickDice extends Dice {
    private int weightedSide;
    private int weight;
    public TrickDice(...) {...}
    public void roll() {...}
    public String toString() {...}
    public int getWSide() {...}
    public int getWeight() {...}
}
```

Make TrickDice a subclass of Dice.

Inheritance

- Allows programmer to derive a class from an existing one
- Existing class is called the parent class, or superclass
- Derived class is called the child class or subclass
- The child class inherits the (public) members defined for the parent class
- Inherited trait can be accessed as though it was locally declared (defined)

Which components get inherited?

- public components get inherited
- private components exist in object of child class, but cannot be directly accessed in child class ⇒ we say they are not inherited
- Note the difference between inheritance and existence!
**protected visibility** (see Sec 4.4)

- Visibility modifiers control which members get inherited
- **private**
  - Not inherited, can be accessed by local class only
- **public**
  - Inherited, can be accessed by all classes
- **protected**
  - Inherited, can be accessed by subclasses
- **Access**: access as though declared locally
- All variables from a superclass exist in the subclass, but some (private) cannot be accessed directly

---

**Inheritance**

Inheritance relationships are shown in a class diagram, with the arrow pointing to the parent class

```
  Dice
   ↓
  TrickDice
```

An **is-a** relationship: the child **is a** more specific version of the parent

**Single** inheritance: one parent only