- Previous Lecture:
  - Selection sort, linear search, binary search *[in section]*
- Today's Lecture:
  - Array of objects
  - Searching in an array of objects
  - Inheritance—extending a class
- Reading:
  - Sec 11.1, 11.2, 11.4, 11.5

Separate classes—each has its own members

```
class Dice {

 private int top;
 private int sides;

 public Dice(…) {…}
 public void roll() {…}
 public String toString(){…}
 public int getTop() {…}
 public int getSides() {…}
}
```

```
class TrickDice {

 private int top;
 private int sides;
 private int weightedSide;
 private int weight;

 public TrickDice(…) {…}
 public void roll() {…}
 public String toString(){…}
 public int getTop(){…}
 public int getSides() {…}
 public int getWSide() {…}
 public int getWeight() {…}
}
```

Can we get all the functionality of Dice in TrickDice without re-writing all the Dice components in class TrickDice?

```
class Dice {

 private int top;
 private int sides;

 public Dice(…) {…}
 public void roll() {…}
 public String toString(){…}
 public int getTop() {…}
 public int getSides() {…}
}
```

```
class TrickDice {

 //everything in class Dice
 //plus new/modified stuff
 //below

 private int weightedSide;
 private int weight;

 public TrickDice(…) {…}
 public void roll() {…}
 public String toString(){…}
 public int getWSide() {…}
 public int getWeight() {…}
}
```

Yes! Make TrickDice a subclass of Dice.

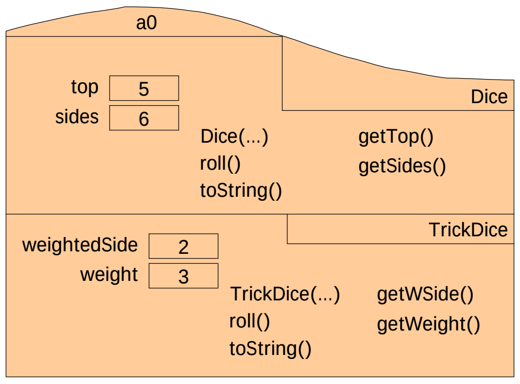
```
class Dice {

 private int top;
 private int sides;

 public Dice(…) {…}
 public void roll() {…}
 public String toString(){…}
 public int getTop() {…}
 public int getSides() {…}
}
```

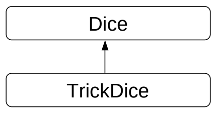
```
class TrickDice extends Dice
{
 private int weightedSide;
 private int weight;

 public TrickDice(…) {…}
 public void roll() {…}
 public String toString(){…}
 public int getWSide() {…}
 public int getWeight() {…}
}
```



Inheritance

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



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

Single inheritance: one parent only

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)*

Calling one constructor from another

- In a subclass' constructor, call the superclass' constructor with the keyword super instead of the superclass' (constructor's) name
- Always make a call to the superclass' constructor as the 1st statement in a constructor in a subclass!

```
class TrickDice extends Dice {

  private int weightedSide;  //Weighted side appears more often
  private int weight;        //Weighted side appears weight
                             //  times as often as other sides

  /** TrickDice has side s appearing with weight w */
  public TrickDice(int numFaces, int s, int w) {
     super(numFaces);
     weightedSide= s;
     weight= w;
  }

  //other methods...
}
```

```
class Dice {
  private int top;    // top face
  private int sides;  // number of sides

  /** A Dice has numSides sides and the top face is random */
  public Dice(int numSides) {
    sides= numSides;
    roll();
  }

  /** top gets a random value in 1..sides */
  public void roll() { setTop(randInt(1,getSides())) ; }

  /** = random int in [low..high], low<high */
  public static int randInt(int low, int high) {
    return (int) (Math.random()*(high-low+1))+low;
  }

  /** Set top to faceValue  */
  private void setTop(int faceValue) { top= faceValue; }
```

Reserved word `super`

Invoke constructor of superclass

```
super(parameter-list);
```

`parameter-list` must match that in superclass' constructor

Calling one constructor from another

- In a subclass' constructor, call the superclass' constructor with the keyword super instead of the superclass' (constructor's) name
- To call another constructor from a constructor in the same class, use the keyword this
- Always make a call to a constructor (super or this) as the 1st statement in a constructor in a subclass!