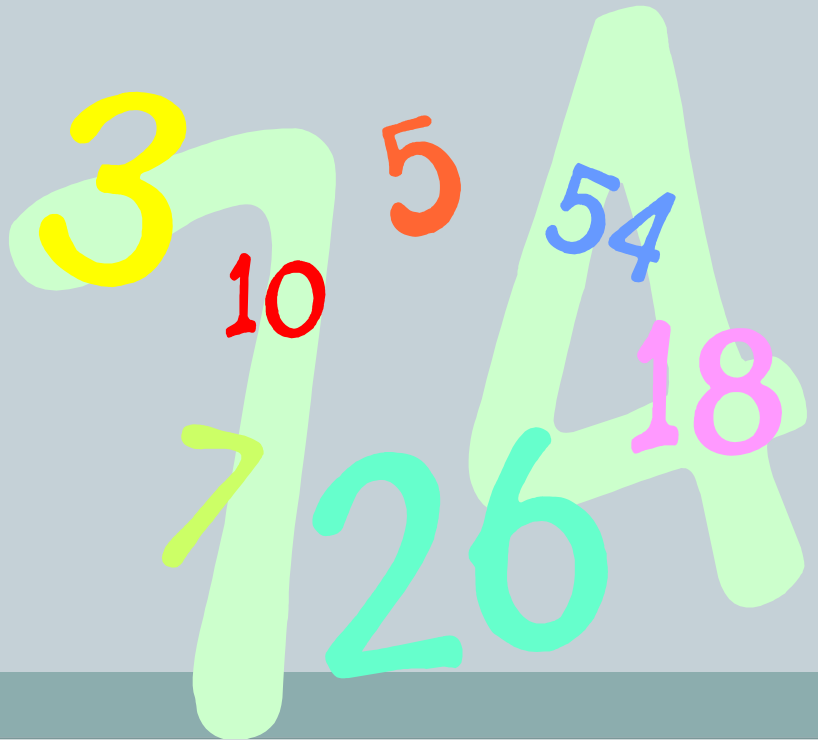
Enumerations

Outline

- Avoiding magic numbers
  - Variables takes on a small set of values
  - Use descriptive names instead of literal values
  - Java enumerations
  - Using in a switch statement



Variables from a Set of Values

- Magic numbers
  - Where did the value come from?
  - What does it mean?
  - What if you mistype the number?
  - What if you want to keep value in specific range?


```
int direction = 0;

...

if ((direction == 1) || (direction == 3) ||
    (direction == 5) || (direction == 7))
{ /* TBD */ }

direction = 0;           // Valid???
direction = 8;           // Valid???
direction = -2729;       // Valid???
```

Variables from a Set of Values

- Solution 1: Create final constants
  - Descriptive names means everybody can read
  - Bugs less likely, typo in name = compile error
  - Keyword `final` ensures nobody can change value

```
final int NORTH      = 0;
final int NORTHEAST  = 1;
final int EAST       = 2;
final int SOUTHEAST  = 3;
final int SOUTH      = 4;
final int SOUTHWEST  = 5;
final int WEST       = 6;
final int NORTHWEST  = 7;

int direction = NORTH;

...

if ((direction == NORTHEAST) || (direction == SOUTHEAST) ||
    (direction == SOUTHWEST) || (direction == NORTHWEST))
{ // TBD }
```

Constants not Always Ideal

```
final int NORTH      = 0;
final int NORTHEAST  = 1;
final int EAST       = 2;
final int SOUTHEAST  = 3;
final int SOUTH      = 4;
final int SOUTHWEST  = 5;
final int WEST       = 6;
final int NORTHWEST  = 7;
```

Problem 1: Tedious to type. Also easy to mess up, e.g. setting two constants to same value.

```
int direction = 0;

...

if ((direction == NORTHEAST) || (direction == SOUTHEAST) ||
    (direction == SOUTHWEST) || (direction == NORTHWEST))
{/* TBD */}

direction = 0;            // Valid???
direction = 8;            // Valid???
direction = -2729;        // Valid???
```

Problem 2: Not forced to use the friendly names.

Problem 3: Not forced to stay in range. What does it mean to be 8 or -2729 if you are a compass direction?

Enumerations

- A better solution: enumerations
  - Specifies exact set of friendly names
  - Compiler ensures we stay in range

Easiest to declare outside class. Semicolon is optional.

```
public enum Compass {NORTH, NORTHEAST, EAST, SOUTHEAST,
                     SOUTH, SOUTHWEST, WEST, NORTHWEST}
public class CompassTest
{
  public static void main(String [] args)
  {
     Compass direction = Compass.NORTH;
     if ((direction == Compass.NORTHEAST) ||
         (direction == Compass.SOUTHEAST) ||
         (direction == Compass.SOUTHWEST) ||
         (direction == Compass.NORTHWEST))
        {/* TBD */}

     direction = 0;
   }
}
```

Now a compile error.
Way to watch our back compiler!

Enumeration Tricks

- Enumerations
  - Actually objects with a few handy methods:

`toString()`	Print out friendly name corresponding to value of variable
`values()`	Returns array of all the possible values type can take on

```
public enum Compass {NORTH, NORTHEAST, EAST, SOUTHEAST,
                     SOUTH, SOUTHWEST, WEST, NORTHWEST}
...
for (Compass d : direction.values())
{
   if (checkMonster(hero, d))
      System.out.println("You see a monster to the " +
                         d.toString());
}
```

for-each loop, goes over all values of the enumeration

switch Statement

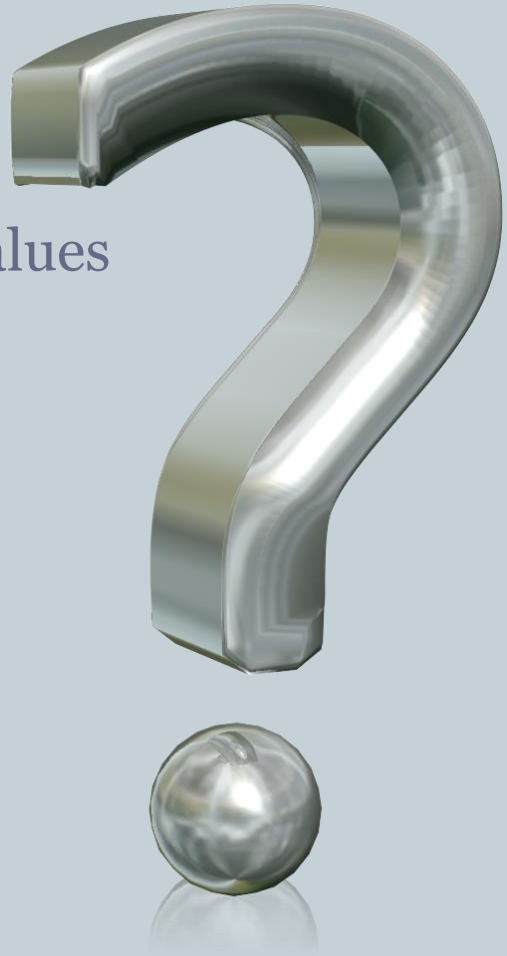
```
Compass direction = Compass.NORTH;

switch (direction)
{
    case NORTH:
        hero.move(0, 1);
        System.out.println("Walking north");
        break;
    case SOUTH:
        hero.move(0, -1);
        System.out.println("Walking south");
        break;
    case EAST:
        hero.move(1, 0);
        System.out.println("Walking east");
        break;
    case WEST:
        hero.move(-1, 0);
        System.out.println("Walking west");
        break;
}
```

Note: normally you need "Compass.", but not in switch case since Java knows type

You can have as many statements as you want between case and break.

Summary

- Avoiding magic numbers
  - Variables takes on a small set of values
  - Use descriptive names instead of literal values
  - Java enumerations
  - Using in a switch statement



Hands On Exercise

- Open Moodle, go to CSCI 136, Section 11
- Open the dropbox for today – In-Class Exercise 4
- Drag and drop your program files to the Moodle dropbox
- You get: 1 point if you turn in something, 2 points if you turn in something that is correct.

- Create an enumeration of possible letter grades named Grade.java
- Create a GradeTest.java program that creates a variable of type Grade and assigns a grade, then use a switch statement to print out an appropriate message. Your switch statement should cover any possible letter grade in case I test it with something other than what you assigned in your program.