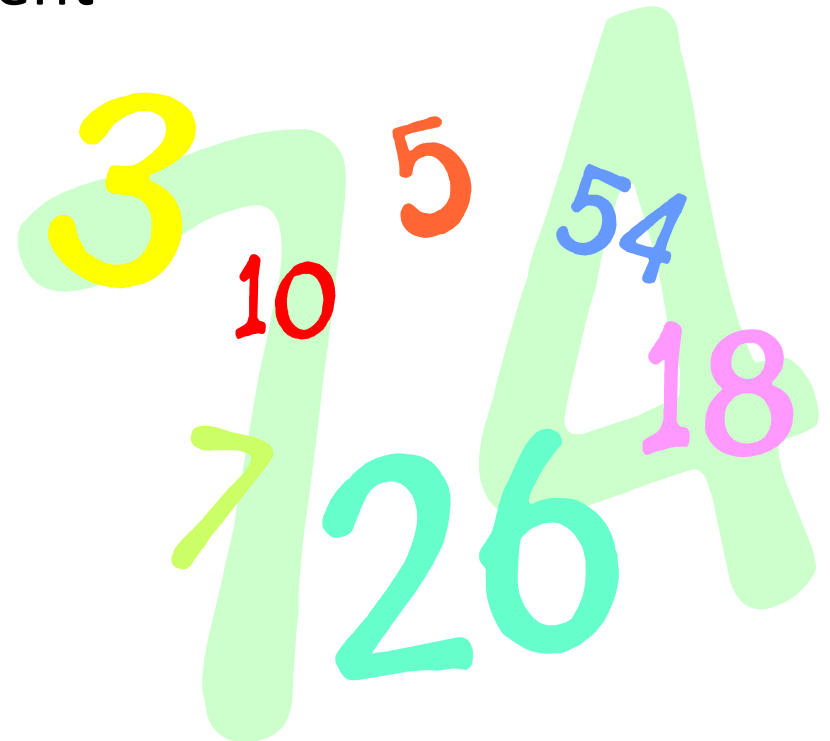


Enumerations



Overview

- 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???
```

- **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;
```

Problem 2: Not forced to use the
friendly names.

```
...
```

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

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

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:

<code>toString()</code>	Print out friendly name corresponding to value of variable
<code>values()</code>	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

- Magic numbers considered harmful!
 - Use Java enumerations instead
 - Descriptive names for what each value means
 - Can be used in a switch statement
 - Can easily loop over all values or print out name

