Comp 110
More Types

Instructor: Sasa Junuzovic
Prerequisites

- Types Math
**Primitive Types**

- int, double, boolean, long, short, float, byte
- char
PRIMITIVE TYPES

- Constants (Literals & Named Constants)
- Operations with Invocation Syntax
Char Constants

char {letters, digits, operations, ...}

16 bits

Escape Sequence

newline character
## Useful Escape Sequences

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Character Denoted</th>
</tr>
</thead>
<tbody>
<tr>
<td>'</td>
<td>'</td>
</tr>
<tr>
<td>\n</td>
<td>new line</td>
</tr>
<tr>
<td>\b</td>
<td>back space</td>
</tr>
<tr>
<td>\</td>
<td>\</td>
</tr>
<tr>
<td>\t</td>
<td>tab</td>
</tr>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
</tbody>
</table>
ORDERING CHARACTERS

“...‘a’...”

Ordinal number (integer code)

Position in ordered character list
ORDERING CHARACTERS

" ... "'a' 'b' 'c' ...
" ... "'A' 'B' 'C' ...
" ... "'0' '1' '2' ...

`'a' > 'b'` false
`'B' > 'A'` true
`'4' > '0'` true

`'a' > 'A'` ???
`'a' > '0'` ???
Converting Between Characters and Their Ordinal Numbers

- `(int) 'a'` leads to `0` as the ordinal number of `a`
- `(char) 55` leads to the character whose ordinal number is 55
- Implicit conversion to wider type

- `(int) ' '` leads to `0`
- `(char) 0` leads to `'
- `(int) 'd'` leads to `???
- `(char) 1` leads to `???
- `(char) -1` (X)

- `(int) 'c' - (int) 'a'` leads to `2`
- `(char) (c' - 2)` leads to `'a'`
- `(char) (A' + 2)` leads to `'C'`
- `(char) (C' - A' + a')` leads to `'c'`
A Useful Character Operation

- Character.isLetter(c)
  - true if c is a letter
- Character.isLetter('c')
  - true
- Character.isLetter('A')
  - true
- Character.isLetter('1')
  - false
- Character.isLetter(' ')  
  - false
**STRING CONSTANTS**

Object Type

String {sequences of characters}

Variable size

- “hello”
- “hello 123”
- ‘a’
- “hello

123”
- “123”
- “a”
- “”
- “\”
- “\”
ACCESSING STRING COMPONENTS

String s = “hello world”;

s.getFirstChar()  
s.getSecondChar()  

StringIndexOutOfBoundsException

Index

s.charAt(0) → ‘h’

s.charAt(1) → ‘e’

s.charAt(-1) → ‘

s.charAt(11) → ‘

s.length() → 11

“ ”.length() → 1

“”.length() → 0
**ACCESSING SUBSTRING**

```java
public String substring (int beginIndex, int endIndex)
```

- `s.substring(beginIndex, endIndex)`
- `→ s.charAt(beginIndex) .. s.charAt(endIndex-1)`

- "hello world".substring(4,7) → "o w"
- "hello world".substring(4,4) → ""
- "hello world".substring(7,4) → StringIndexBounds Exception
**Changing Strings?**

Stings are read-only (immutable)

```
"hello" + "world"
```

"hello world"

Three different instances
Useful String Operations

- `s.toLowerCase()` → copy of `s` with letters converted to lower case
- `s.toUpperCase()` → copy of `s` with letters converted to upper case
- “Hello World”.toLowerCase() → “hello world”
- “Hello World”.toUpperCase() → “HELLO WORLD”
CLASSIFICATION

- Underweight
- Normal
- Overweight
- Obese
- Extremely Obese

Data type to store a value capturing the BMI classification?

- African Indian
- American Indian
- Asian
- Native Hawaiian
- White
- Some Other Race

Data type to store a value capturing the Race?
**Int Constants in Separate Interfaces**

```java
public interface IntBMIClassification {
    public static int UNDER_WEIGHT = 0;
    public static int NORMAL = 1;
    public static int OVER_WEIGHT = 2;
    public static int OBESE = 3;
    public static int EXTREMELY_OBESE = 4;
}

public interface IntRace {
    public static int AFRICAN_AMERICAN = 0;
    public static int AMERICAN_INDIAN = 1;
    public static int ASIAN = 2;
    public static int NATIVE_HAWAIIAN = 4;
    public static int WHITE = 4;
    public static int SOME_OTHER_RACE = 5;
}
```
IMPLEMENTATION WITH INT CONSTANTS

```java
int race = IntRace.AFRICAN_AMERICAN;
pUBLIC int getRace() {
   return race;
}
public void setRace(int newVal) {
   race = newVal;
}
pUBLIC int getBMIClassification() {
    double bmi = getBMI();
    if (bmi <= MAX_UNDERWEIGHT_BMI) {
        return BMIIIntClassification.UNDER_WEIGHT;
    } else if (bmi <= MAX_NORMAL_BMI) {
        return BMIIIntClassification.NORMAL;
    } else if (bmi <= MAX_OVERWEIGHT_BMI) {
        return BMIIIntClassification.OVER_WEIGHT;
    } else if (bmi <= MAX_OBESE_BMI) {
        return BMIIIntClassification.OBESE;
    } else {
        return BMIIIntClassification.EXTREMELY_OBESE;
    }
```
**PROGRAMMER ERRORS POSSIBLE**

```java
public interface BMIIntClassification {
    public static int UNDER_WEIGHT = 0;
    public static int NORMAL = 1;
    public static int OVER_WEIGHT = 2;
    public static int OBESE = 3;
    public static int EXTREMELY_OBESE = 3;
}
```

Programming language does not know the relationship among constants!
User can make mistake in assigning a value
public interface StringBMIClassification {
    public static String UNDER_WEIGHT = "Under Weight";
    public static String NORMAL = "Normal";
    public static String OVER_WEIGHT = "Over Weight";
    public static String OBESE = "Obese";
    public static String EXTREMELY_OBESE = "Extremely Obese";
}

public interface StringRace {
    public static String AFRICAN_AMERICAN = "African American";
    public static String AMERICAN_INDIAN = "American Indian";
    public static String ASIAN = "Asian";
    public static String NATIVE_HAWAIIAN = "Native Hawaiian";
    public static String WHITE = "White";
    public static String SOME_OTHER_RACE = "Some Other Race";
}
**IMPLEMENTATION WITH STRING CONSTANTS**

```java
String race = StringRace.AFRICAN_AMERICAN;
public String getRace() {
    return race;
}
public void setRace (String newVal) {
    race = newVal;
}
public String getBMIClassification() {
    double bmi = getBMI();
    if (bmi <= MAX_UNDERWEIGHT_BMI)
        return StringBMIClassification.UNDER_WEIGHT;
    else if (bmi <= MAX_NORMAL_BMI)
        return StringBMIClassification.NORMAL;
    else if (bmi <= MAX_OVERWEIGHT_BMI)
        return StringBMIClassification.OVER_WEIGHT;
    else if (bmi <= MAX_OBESE_BMI)
        return StringBMIClassification.OBESE;
    else
        return StringBMIClassification.EXTREMELY_OBESE;
}
```
Errors Possible

String type is space inefficient

User or programmer can make mistake in assigning a value
**String Issues**

User or programmer can make mistake in assigning a value

String type is space inefficient
**Enum**

- Like a string set, a set of named choices
- Like an int set, each choice is a number
Like class or an interface, an enum is declared in its own file.
### IMPLEMENTATION WITH STRING CONSTANTS

```java
EnumRace race = EnumRace.AFRICAN_AMERICAN;
public EnumRace getRace() {
    return race;
}
public void setRace (EnumRace newVal) {
    race = newVal;
}
public EnumBMIClassification getBMIClassification() {
    double bmi = getBMI();
    if (bmi <= MAX_UNDERWEIGHT_BMI)
        return EnumBMIClassification.UNDER_WEIGHT;
    else if (bmi <= MAX_NORMAL_BMI)
        return EnumBMIClassification.NORMAL;
    else if (bmi <= MAX_OVERWEIGHT_BMI)
        return EnumBMIClassification.OVER_WEIGHT;
    else if (bmi <= MAX_OBESE_BMI)
        return EnumBMIClassification.OBESE;
    else
        return EnumBMIClassification.EXTREMELY_OBESE;
}
```
DISPLAYING AND EDITING ENUM