Data Conversion

- Implicit
  - Assignment conversion

    ```
    int a = 5;
    double b = a;
    ```

  - Promotion

    ```
    int a = 5;
    double b = 2.0;
    c = b + a;  // a is promoted to double
    ```

- Explicit (Type casting)

    ```
    int a = 7;
    double b = (double)a;
    ```

Data Conversion

- Widening conversion

```
int a = 8;
double b = a;           // Implicit
double c = (double)a;   // Explicit
```

- Narrowing conversion

```
double a = 7.8;
int b = (int)a; // Always explicit
```

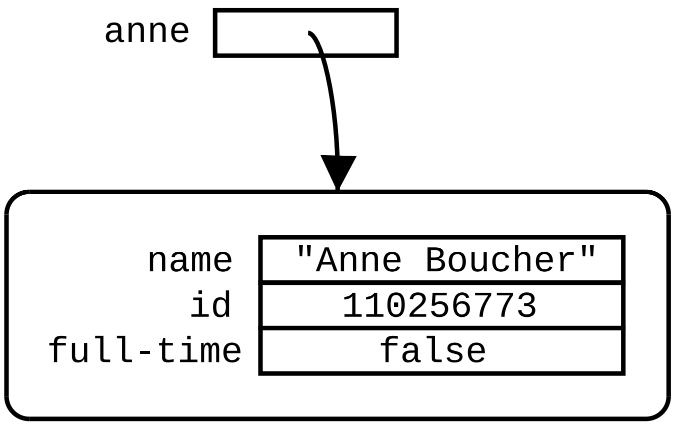
- Narrowing conversion can result in loss of information
- Sometimes widening conversions are needed to ensure a particular type of operation:

```
int a = 7;
double b = a / 2;
double c = (double)a / 2;
```

McGill

Objects and Classes

- Programs manipulate data
- Variables store data
- A variable holds either:
  - a value from a primitive data type (int, boolean, char, ...)
  - or a reference to an object
- An *object* is a composite piece of data: it is a group of variables treated as a unit



Objects and classes

- The data type of an object is a *class*
- Classes have *methods*
- Methods are the operations of a class
- Applying a method to an object is written:

  `objectreference.methodname(parameters)`

  where `methodname` is defined in the class of the object
- For example:

  ```
  anne.change_id(260298776)
  ```

Strings

- `String` is a class, and particular strings are objects
- Methods of the `String` class define operations on strings
  - `int length()`
  - `char charAt(int index)`
  - `String substring(int offset, int endIndex)`
  - `boolean equals(String s)`
  - `String concat(String s)`
  - `String replace(char a, char b)`

Strings

- Examples of `int length()`

```
String question;
int l;
question = “Is this course  easy?”;

l = question.length();

System.out.println(l);   // 21

String answer;
answer = “It depends...”;

l = answer.length();

System.out.println(l);   // 13

String very_short_message = “”;
System.out.println( very_short_message.length() );
```

Strings

- Examples of `char charAt(int index)`

```
String phrase;
char initial1, initial2, initial3,
     initial4, initial5;
String acronym;

phrase = ‘‘Emacs makes a computer swell’’;

initial1 = phrase.charAt(0);
initial2 = phrase.charAt(6);
initial3 = phrase.charAt(12);
initial4 = phrase.charAt(14);
initial5 = phrase.charAt(23);

acronym = ‘‘’’ + initial1 + initial2
               + initial3 + initial4 + initial5;
```

Strings

- The argument or parameter of `charAt` can be any integer expression

```
String phrase;
char c;
int start = 3;

phrase = “Strings do not have to make sense.”;

c = phrase.charAt( start + 2 );

// c == ‘g’

c = phrase.charAt( phrase.length() - 1 );

// c == ‘.’

c = phrase.charAt( phrase.length() );
// Runtime error
```

Strings

- Since the `charAt` method returns a character, it can be used in any character expression, and in particular it can be used within string expressions

```
String word1 = ‘‘rat’’, word2 = ‘‘case’’;
String word3;
word3 = word1 + word2.charAt(2);

// word3 contains ‘‘rats’’
```

Strings

- `charAt` cannot be used to modify a string

```
String word = “clap”;
word.charAt(0) = ‘f’; // WRONG!
```

- Strings in Java are immutable: they cannot change
- But String references can change:

```
String word = “clap”;
String new_word;
new_word = “f” + word.charAt(1)
         + word.charAt(2) + word.charAt(3);
word = new_word;

// word contains “flap”;
```

Strings

- Examples of
  `String substring(int offset, int endIndex)`

```
String word = “clap”;
String end, new_word;
end = word.substring(1, 4);

// end contains “lap”;

new_word = “f” + end;

// new_word contains “flap”
```

Strings

- `s.substring(i, j)` returns the part of string `s` beginning at index `i` and ending at index `j-1`

```
String phrase, subject, verb, article, noun;

phrase = “This is a string”;
subject = phrase.substring(0, 4);
verb = phrase.substring(5, 7);
article = phrase.substring(8, 9);
noun = phrase.substring(10, phrase.length());

System.out.println(subject+article+noun+verb);

// Prints
// Thisastringis
```

Strings

- Since the `substring` method returns a `String`, it can be used within any string expression

```
String old_phrase = “This is a string”;
int size = old_phrase.length();
String new_phrase;

new_phrase = old_phrase.substring(0, 8)
             + “not ”
             + old_phrase.substring(8, size);

// new_phrase contains “This is not a string”
```

Strings

- Examples of `boolean equals(String s)`

```
String pet1 = “cat”, pet2 = “rat”;
String end1, end2;
boolean same_pet, same_end;

same_pet = pet1.equals(pet2);

end1 = pet1.substring(1, pet1.length() );
end2 = pet2.substring(1, pet2.length() );

same_end = end1.equals(end2);
```

- For every pair of strings `a` and `b`, `a.equals(b)` returns the same as `b.equals(a)`

Strings

- Since the `equals` method returns a `boolean`, it can be used in any boolean expression

```
String season = “Winter”;
float temp = -5.0f;
boolean warm;

warm = !season.equals(“Winter”) || temp >= -10.0f;
```

season.equals(“Winter”)	temp>=-10.0f	!season.equals("Winter")	warm
true	true	false	true
true	false	false	false
false	true	true	true
false	false	true	true

Strings

- Examples of `String concat(String s)`

```
String sentence;
sentence = “This sentence is ”;
sentence = sentence.concat(“ false”);
```

- If `a` and `b` are strings, `a + b` is shorthand for `a.concat(b)`

Strings

- Examples of `String replace(char a, char b)`

```
String message, encoded;
message = “This message is irrelevant”;
encoded = message.replace(‘e’, ‘x’);

// encoded contains “This mxssagx is  irrxlxvant”

encoded = encoded.replace(‘a’, ‘y’);
encoded = encoded.replace(‘i’, ‘z’);
encoded = encoded.replace(‘r’, ‘w’);
encoded = encoded.replace(‘s’, ‘u’);
encoded = encoded.replace(‘ ’, ‘-’);
encoded = encoded.replace(‘t’, ‘v’);

// encoded contains “Thzu-mxuuygx-zu--zwwxlxvynv”
```

An example

- Problem: Given a four letter word, print the word in reverse.
- Analysis:
  - Information involved: a four letter word, *w*.
  - Input: *w*
  - Output: a word *v* which is the reverse of *w*
  - Definitions:
    - The *reverse* of a word *w* is a word *v* which has the the same characters as *w*, but in inverse order: the first letter of *v* is the last of *w*, the second letter of *v* is the second-to-last of *w*, etc.
  - Restrictions: *w* is assumed to have only four letters

An example

- Design

1. Obtain the word *w*
2. Create a new word *v*, initially empty
3. Add the last character of *w* to the end of *v*
4. Add the third character of *w* to the end of *v*
5. Add the second character of *w* to the end of *v*
6. Add the first character of *w* to the end of v
7. Print *v*

An example

- Implementation

```
import cs1.Keyboard;
public class Reverse {
  public static void main(String[] args)
  {
    String w, v;

    System.out.print(‘‘Enter a four letter word: ’’)
    w = Keyboard.readString();

    v = ‘‘’’;
    v = v + w.charAt( 3 );
    v = v + w.charAt( 2 );
    v = v + w.charAt( 1 );
    v = v + w.charAt( 0 );

    System.out.println(v);
  }
}
```

An example

- Implementation

```
import cs1.Keyboard;
public class Reverse {
  public static void main(String[] args)
  {
    String w, v;

    System.out.print(‘‘Enter a four letter word: ’’)
    w = Keyboard.readString();

    v = ‘‘’’ + w.charAt( 3 ) + w.charAt( 2 )
        + w.charAt( 1 ) + w.charAt( 0 );

    System.out.println(v);
  }
}
```

An example

- Design (alternative)

1. Obtain the word *w*
2. Create a new word *v*, initially empty
3. Add the first character of *w* to the front of *v*
4. Add the second character of *w* to the front of *v*
5. Add the third character of *w* to the front of *v*
6. Add the last character of *w* to the front of v
7. Print *v*

An example

- Implementation

```
import cs1.Keyboard;
public class Reverse {
  public static void main(String[] args)
  {
    String w, v;

    System.out.print(‘‘Enter a four letter word: ’’)
    w = Keyboard.readString();

    v = ‘‘’’;
    v = ‘‘’’ + w.charAt( 0 ) + v;
    v = ‘‘’’ + w.charAt( 1 ) + v;
    v = ‘‘’’ + w.charAt( 2 ) + v;
    v = ‘‘’’ + w.charAt( 3 ) + v;

    System.out.println(v);
  }
}
```

An example

- Implementation

```
import cs1.Keyboard;
public class Reverse {
  public static void main(String[] args)
  {
    String w, v;

    System.out.print(“Enter a four letter word: ”)
    w = Keyboard.readString();

    v = ‘“’ + w.charAt( 3 ) + w.charAt( 2 )
        + w.charAt( 1 ) + w.charAt( 0 );

    System.out.println(v);
  }
}
```

Another example

- Problem: Given a four letter word, determine whether the word is a palindrome

- Analysis:
  - Information involved: a four letter word, *w*.
  - Input: *w*
  - Output: true if the word is a palindrome, false otherwise
  - Definitions:
    - A word is a *palindrome* if it is the same as its own reverse, e.g. (noon, radar, wow, pop, 2002, ...)
  - Restrictions: *w* is assumed to have only four letters

Another example

- Design:

1. Obtain word *w*
2. Compute the reverse of *w*: let *v* be the reverse of *w*
3. Compare *v* and *w*. Let *result* be true if *w* and *v* are equal, and false otherwise.
4. Print *result*

Another example

```
import cs1.Keyboard;
public class Palindromes {
  public static void main(String[] args)
  {
    String w, v;
    boolean result;

    System.out.print(‘‘Enter a four letter word: ’’)
    w = Keyboard.readString();

    v = ‘‘’’;
    v = v + w.charAt(3);
    v = v + w.charAt(2);
    v = v + w.charAt(1);
    v = v + w.charAt(0);

    result = v.equals(w);

    System.out.println(result);
  }
}
```

McGill

Another example

- Design (alternative):

1. Obtain word *w*

2. Compare the first character of *w* with its last character and the second character with the thirs character. Let result be true if both comparisons yield true, and false otherwise.

3. Print *result*

Characters

- Values of the `char` data type can be compared using the traditional relational operators:

```
char a = 'P', b = 'Q';
boolean c, d, e, f, g, h;
c = a == b;    // c == false
d = a != b;    // d == true
e = a < b;     // e == true
f = a > b;     // f == false
g = a <= b;    // g == true
h = a >= b;    // h == false

char a = 'Q', b = 'Q';
boolean c, d, e, f, g, h;
c = a == b;    // c == true
d = a != b;    // d == false
e = a < b;     // e == false
f = a > b;     // f == false
g = a <= b;    // g == true
h = a >= b;    // h == true
```

Another example

```
import cs1.Keyboard;
public class Palindromes {
  public static void main(String[] args)
  {
    String w;
    boolean result;

    System.out.print(‘‘Enter a four letter word: ’’)
    w = Keyboard.readString();

    result = w.charAt(0) == w.charAt(3)
           && w.charAt(1) == w.charAt(2);

    System.out.println(result);
  }
}
```