CMPSC 100

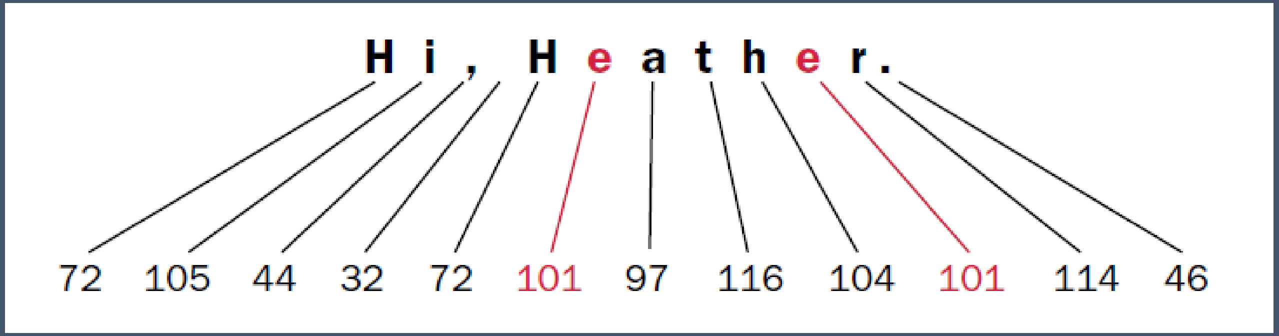
Computational Expression

Recap

Primitives

Type	Storage	Min Value	Max Value
byte	8 bits	–128	127
short	16 bits	–32,768	32,767
int	32 bits	–2,147,483,648	2,147,483,647
long	64 bits	–9,223,372,036,854,775,808	9,223,372,036,854,775,807
float	32 bits	Approximately –3.4E+38 with 7 significant digits	Approximately 3.4E+38 with 7 significant digits
double	64 bits	Approximately –1.7E+308 with 15 significant digits	Approximately 1.7E+308 with 15 significant digits

Strings as non-primitive



Primitives aren’t “type-cast”

```
/** The entry point.
 *
 * @param args The command line arguments
 */
public static void main(String[] args) {
     int a = Integer.parseInt(args[0]);
}
```

Data conversion

- Because String type variable aren’t primitive, the capital “S” in the type indicates that it comes from a library (due to naming convention)
  - String is part of the JAVA API—core functionality packaged with Java releases to make them viably useful
- Similarly, we have parts of the API like Integer or Double
  - These provide services that allow us to convert one primitive type to another with additional functionality

A simple problem

```
String count = “6”;
int sum = 600;
int average = sum / count
```

ERROR!

Data conversion

- Variables acquire/inherit types four ways:
  - Assignment
    ```
    int count = 6;
    ```
  - Promotion
    ```
    double count = 11.0;
    double average = sum / count;
    ```
  - Casting
    ```
    int count = 11;
    double average = sum / (double)count;
    ```
  - Parsing
    - `String number = args[0]; // args[0] = “6”`
    - `int count = Integer.parseInt(number);`

Arguments and input

```
/** The entry point.
 *
 * @param args The command line arguments
 */
public static void main(String[] args) {
     int a = Integer.parseInt(args[0]);
}
```

Arguments and input

```
gradle run --args=“100”
==
java JavaProgram 100
```

Arguments and input

(There is.)

- "There has to be a different way."
- Additional packages in Java give us the ability to get input in formats we can expect/prepare for.
  - This part of the universe is called: "java.util.Scanner"
    - This allows us to gather inputs from users using various sources.

```
java.lang.System.out.println("");
```

Classes

```
java.util.Scanner;
```

Activity

- “cd” to your class-activites repository
- Perform a “git pull download master”
- Open the file “AddressLabel.java” in the src/main/java/snailmail directory
- Take a minute to browse the code before moving on:
  - What is familiar?
  - What is new?

Activity

```
// Set up Scanner to take from System.in
Scanner scan = new Scanner(System.in);
/*
 * Create prompts by printing text requested,
 * then implementing Scanner.
 */
System.out.print("Enter building number: ");
int buildingNumber = scan.nextInt();
/*
 * nextInt() doesn't consume the next line character ("\n"),
 * so we ask the scanner to move along to the next line without
 * assigning the input to a variable. In essence, it creates
 * "garbage" out of it.
 */
scan.nextLine();
```

Creates a new instance of an object

Uses a method from that object

Uses a method from that object

Activity

```
System.out.print("Enter street name: ");
String streetName = scan.nextLine();
System.out.print("Enter city: ");
String cityName = scan.nextLine();
System.out.print("Enter two-letter state: ");
String stateAbbrev = scan.nextLine();
System.out.print("Enter zip code: ");
// TODO: Figure out what goes here: String, int,
other?
```

Activity

Finish the remainder of the request yourself!

Test using:

```
gradle -q --console plain run
```

in the 23-September folder