Learning Outcomes

Upon completing this assignment, students should be able to

- Use the *repeated division* method to convert a Java `int` into a `String` representation in *binary*, *decimal*, or *hexadecimal*.
- Test and document the code.

Introduction

Converting between number bases is a skill that is necessary when working at a low-level in a computer. Tools will display memory and register values in hex, decimal, or binary and one must be comfortable with converting these to a common base to be able to check the correctness of a circuit.

One way to learn how to do this is to practice rote translation across one or two hundred problems. An alternative, that is designed to get the *algorithm* to stick, is to implement a program, in a language that you know, that does it. Designing, programming, testing, and debugging such a program will make a lasting impression in your mind. Using Java, a programming language you are comfortable with, makes it easier to focus on the base conversion rather than the implementation details.

This also turns out to be a really good place to start discussing how the `Integer.decode` (and `Integer.parseInt`) method works.

Method

Getting Started

You are to write a Java program that

1. Reads its command-line for arguments
2. Handles commands to generate output in binary, decimal, and/or hex.
3. Handles an arbitrary number of string representations of integers as input values (decimal or hex).
4. Handles a command to show its work.

`main`: construct and `run`

The program design is constrained in that it **must** make good use of methods and **must not** have any `static` methods other than `main`.

An easy way to handle this is to construct an object in `main` (of the application class type) and then call a `run` method on it to do the real work. An example `Hello` program is below:

```
public class Hello {
  private static final String DEFAULT_NAME = "World";
  private String toWhom;

  public Hello(String whom) {
    toWhom = whom;
  }

  public void run() {
    System.out.printf("Hello, %s!\n", toWhom);
  }

  public static void main(String[] args) {
    String whoToSayHelloTo = DEFAULT_NAME;
    if (args.length > 0)
      whoToSayHelloTo = args[0];

    Hello applicationObject = new Hello(whoToSayHelloTo);
    applicationObject.run();
  }
}
```

The key is to note the constructor and the `run` method and then the last two lines of `main`: a `Hello` object is constructed with the provided command-line arguments *or* default values. Then that object is `run` to do the body of the program.

Your program will have more complex argument processing, more constructor parameters, and (hopefully) many more small, single purpose functions to help `run`.

Input/Output

Command-line Parameters

The program takes any number of command-line parameters. An argument string starting with `--` is a command to the program to change its behavior; any other argument is expected to be a string containing an integer, either decimal, `2564`, or hexadecimal, `0x12345`.

There are four recognized commands (any other `--` command should generate a warning message and be ignored).

`--dec` Display the *decimal* translation of the integers.

`--hex` Display the *hexadecimal* translation of the integers.

`--bin` Display the *binary* translation of the integers.

`--verbose` Show the repeated divisions used to generate each translation before showing the translation.

The program should *require* at least one integer to convert.

Example Runs

```
$ java NumberBases --bin 100
  0b110 0100

$ java NumberBases --hex --verbose 25 0x100
  16 |    25
     |     1  9
     |     0  1
  0x19

  16 |   256
     |    16  0
     |     1  0
     |     0  1
  0x100

$ java NumberBases --dec --verbose 185
  10 |    185
     |     18  5
     |      1  8
     |      0  1
  185

$ java NumberBases 0x40
  64
```

Testing

Test program by checking known values. Check edge conditions.

Valid range for unsigned, decimal integers is 0 – 2147483647 $2^{31} - 1$

Documentation

`README`

Must document how you tested. How do you know that it is right?

Must include instructions on how to **compile** and how to **run** the program as submitted. You can test these if you `clone` the repository you submitted into a **brand new location**. You can then follow the instructions and see if all the necessary files are in the repo.

Deliverables

Submission medium: `git` to Gitea at `cs-devel.potsdam.edu`. The repo name is `p001-NumberBases` and it goes in your organization for turning in assignments in CIS 356 this semester.