CISC 3120

C03: Objects, References, and Primitives

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Recap and issues
- Review some constructs for flow control
    - selection & iteration
- Discuss some concepts in Objected-Oriented Programming
- Discuss primitives and references
- Assignments

What did we learn from BeerSong.java?

- Anatomy of a Java class
  - What goes in a Java source code file, what goes in a Java class, and what goes in a method?
  - Where is the entry point of a Java program?
- A few data types
- Identifiers
- Simple and compound statements
- A few flow controls
- Comment
- Java build-in classes (Java libraries)
- Coding style

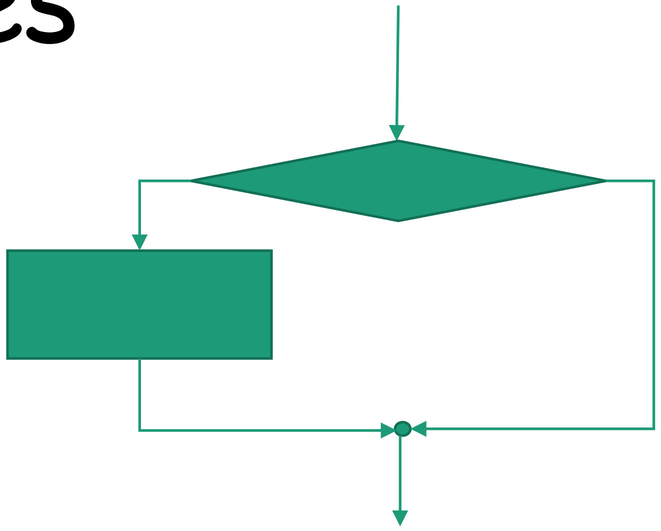
Using Command Line Arguments

- public static void main(String[] args)
  - An array of String objects passed to the main method
- How do we use it?
  - Example: use it to change BeerSong's behavior.

Selection Structures

- Similar to C++
- The if statement
  - The if-then statement
  - The if-then-else statement
- The switch statement (discuss later in C05)

If-Then: Examples



- Example

```
if ( isMoving ) {
        currentSpeed --;
}
```

- Question: which one of the two are legal or illegal in Java and in C++, respectively?

```
if ( 1 )
    currentSpeed --;
}
```

```
if ( true )
    currentSpeed --;
}
```

If-Then: Question

- In Java

```
if ( 1 )
    currentSpeed --;
}
```

✗

```
if ( true )
    currentSpeed --;
}
```

✓

- In C++

```
if ( 1 )
    currentSpeed --;
}
```

✓

```
if ( true )
    currentSpeed --;
}
```

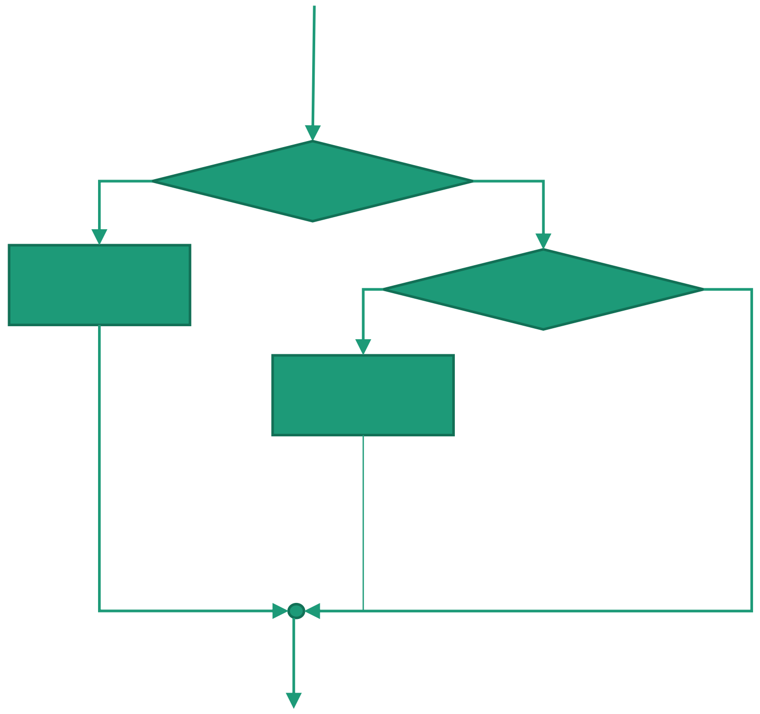
✓

- What can you conclude?

If-Then-Else

- Example

```
if (testscore >= 90) {
    grade = 'A';
} else if (testscore >= 80) {
    grade = 'B';
} else if (testscore >= 70) {
    grade = 'C';
} else if (testscore >= 60) {
    grade = 'D';
} else {
    grade = 'F';
}
```

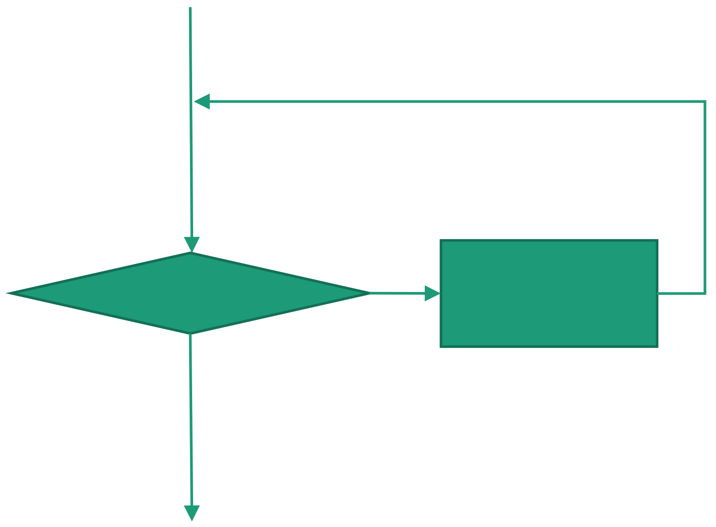


Iterations

- The while statement
- The for statement
  - The basic for statement
  - The enhanced for statement (discuss later in C05)
- The do statement (discuss later in C05)

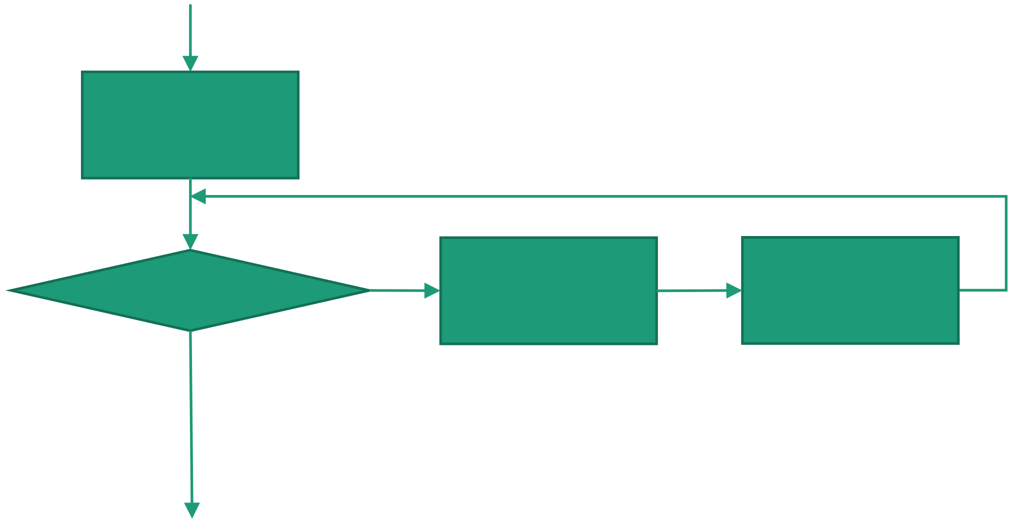
The while Statement

- while (expression) statement
- Example
    - BeerSong.java



The Basic for Statement

- The basic for statement
  - for ([ForInit] ; [Expression] ; [ForUpdate]) Statement



The basic for Statement: Examples

- Example 1

```
for (int i=99; i>=0; i--) {
        System.out.println(i + “bottles of beers on the wall”);
}
```

- Example 2

```
// print out command line arguments
for (int i=0; i<args.length; i++) {
        System.out.println(args[i])
}
```

Questions?

- Flow controls
  - Selections
  - Iterations

Classes and Objects

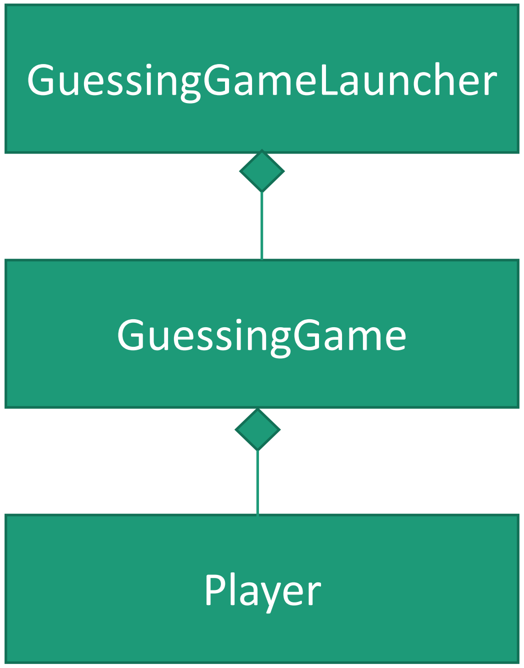
- Divide an application into multiple classes
- Instantiate objects from classes
- Thinking: client & server
  - Client & server interact via method invocation.
    - A client invokes the server's method
  - Some literature call this "message passing".

The Guessing Game

- Simulate a game where 3 players guess a number that is being held as a secrete.
  - Generate a list random numbers.
  - Have 3 players to make a guess.
  - See who makes correct guess.

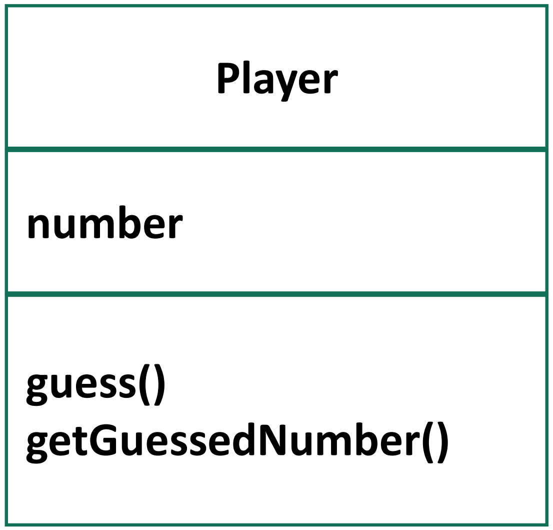
Design of the Guessing Game Application

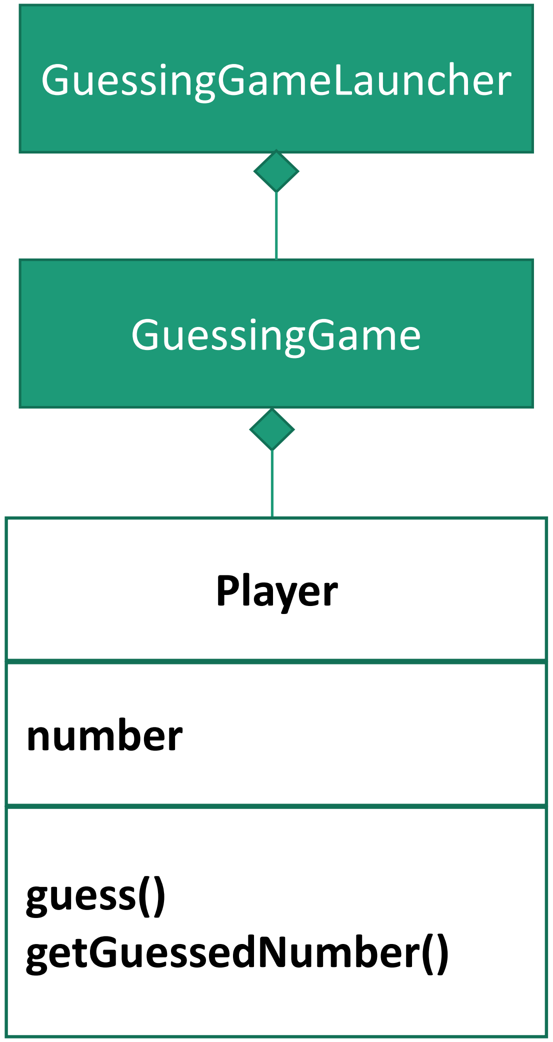
- Divide the application into 3 classes
    - GuessingGameLauncher
    - GuessingGame
    - Player
- The "Composition" pattern



Describing a Class

- UML: Class diagram





Creating Objects

- Create objects (or instances) from a class (or instantiate a class)
  - Using the “new” operator
  - Examples
    - GuessingGame game = new GuessingGame();
    - Player player = new Player();

Object-to-Object Communication

- Method invocation
    - Client & server
    - Client object calls the server object's method
    - The client pass a message to the server
- Example
    - player.guess()
    - player is the server
    - The object that has the statement is the client

Questions?

- Class and objects
- The “composition” pattern
- Object-to-object communication

References

- Everything is an object in Java (except primitives)
- Variables hold references to objects

Object and Reference

- Example
  - GuessingGame game = new GuessingGame();
- GuessingGame: class
- game: variable
- Variable game holds the reference to the object created by “new GuessingGame()”.
- “game” is not, “game” does not hold the object

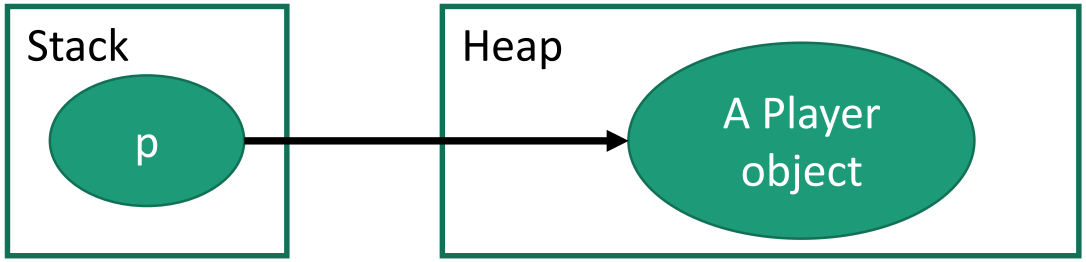
Where are the Objects?

- Player p = new Player();
- Where is the player object?

Where are the Objects?

- JVM memory
  - Stack
    - Where local variables (a.k.a., stack variables) are allocated
  - (Garbage-Collection) Heap
    - Where objects are allocated (note: instance variables are part of an object)

```
public void startGame() {
    Player p = new Player();
}
```



Life Cycle of Objects

- How do I "destroy" the object and release the memory?

Compare it with C++

Java Garbage Collector

- A program runs on the Java Virtual Machine (JVM)
  - Implements automatic memory management
  - Look for objects that are not being used by applications any more, and remove the objects, and freeing the memory.
- In Java, the garbage collector does the memory management for you.
- In C++, you needs to perform memory management all by yourself (using the new and delete operators)

Primitive Data Types

- Special data types built into the language
- Not objects created from a class
- Java has 8 primitive data types

Java Primitive Data Types

- 8 primitive data types

Type	Description	Default	Size	Example Literals
boolean	True or false	False	1 bit	true, false
byte	integer	0	8 bits	(none)
char	Unicode character	\u0000	16 bits	‘a’, ‘u0041’, ‘\101’
short	Integer	0	16 bits	(none)
int	Integer	0	32 bits	-9, -8, 0, 1 2
long	Integer	0	64 bits	3L, 1L, -1L, -3L
float	Floating point	0.0	32 bits	3.14e10f, -1.23e-100f
double	Floating point	0.0	64 bits	1.1e1d, -3.14e10d

Numerical Literals

- A few types: byte, short, int, long, float, double
- Java 7 or newer allow "_" in numerical literals
  - long creditCardNumber = 1234_5678_9012_3456L;
  - long socialSecurityNumber = 999_99_9999L;
  - float pi = 3.14_15F;
  - long hexBytes = 0xFF_EC_DE_5E;
  - long hexWords = 0xCAFE_BABE;
  - long maxLong = 0x7fff_ffff_ffff_ffffL;
  - byte nybbles = 0b0010_0101;
  - long bytes = 0b11010010_01101001_10010100_10010010;
- Prefixes: 0x and 0b indicate hexadecimal and binary values, respectively
- Suffixes: L and F indicate long and float values, respectively

Choose Primitive Data Type

- Require that you understand the needs of your application
  - Examples
    - Do you need a variable to hold whole numbers? What are the range of the whole numbers?
    - If your numbers may have fractions, do they need to be precise?
      - May BigDecimal be more appropriate?

Characters

- Always use single quote for character
- Java character holds a Unicode character
  - A character is a 16-bit Unicode
  - A character literal can be a “Unicode escape”
    - ‘\u00ed’ (í in Spanish)
    - ‘\u00f1’ (ñ in Spanish)

Special Characters

- A few special escape sequences for char and String literals
    - \b (backspace)
    - \t (tab),
    - \n (line feed)
    - \f (form feed)
    - \r (carriage return)
    - \" (double quote)
    - \' (single quote)
    - \\ (backslash)
- Example
    - char c = ‘\b’;

Java Variables

- 4 kinds of variables
  - Instance variables (non-static fields)
  - Class variables (static fields)
  - Local variables
  - Parameters

4 Kinds of Variables: Example

- Identify 4 kinds of variables

```
class Bus {
  static int numOfWheels = 4;
  double speed;

  void accelerate(double acceleration, double duration) {
     double speedIncrement = acceleration * duration;
     speed += speedIncrement;
  }
}
```

4 Kinds of Variables: Example

- Identify 4 kinds of variables

```
class Bus {
  static int numOfWheels = 4;
  double speed;

  void accelerate(double acceleration, double duration) {
     double speedIncrement = acceleration * duration;
     speed += speedIncrement;
  }
}
```

Class variable

Instance variable

Parameters

Local variable

Variable Names

- Variable names are case-sensitive.
  - An unlimited-length sequence of Unicode letters and digits
  - Must begin with a letter, the dollar sign "$", or the underscore character "_".
- Naming convention
  - If not constants (not “final”)
    - Always start with a letter
    - First word are all lower case letters
    - Capitalize first letter of each subsequent words
  - If constants (final)
    - Capitalizing every letter and separating subsequent words with the underscore character

Variable Initialization and Default Values

- Java compiler initializes instance variables with default values
- Java compiles does not initialize local variables
  - Accessing an uninitialized local variable will result in a compile-time error.

Declare Variables of Primitive Types

- Declaration without initialization
  - Examples
    - int count;
    - boolean isDone;
    - double gpa;
- Declaration with initialization
  - Examples
    - int count = 0, sum = 0;
    - boolean hasVisited = false;
    - double gpa = 0.0;

Operators

- Arithmetic operators
- Unary operators
- Equality and Relational operators (discussed in more details later)
- Conditional operators (discussed in more details later)
- Bitwise and bit shift operators (discussed in more details later)

Arithmetic Operators

Operator	Description
+	Additive operator (also used for String concatenation)
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator

Unary Operators

Operator	Description
+	Unary plus operator; indicates positive value (numbers are positive without this, however)
-	Unary minus operator; negates an expression
++	Increment operator; increments a value by 1
--	Decrement operator; decrements a value by 1
!	Logical complement operator; inverts the value of a boolean

Equality and Relational Operators

Operator	Description
==	equal to != not equal to > greater than >= greater than or equal to < less than <= less than or equal to
!=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Conditional Operators

- &&: Conditional-AND
- ||: Conditional-OR
- exhibit "short-circuiting" behavior

Bitwise and Bit Shift Operators

Operator	Description
~	A unary operator that inverts a bit pattern
<<	Signed left-shift operator
>>	Signed right-shift operator
&	Bitwise AND operator
|	Bitwise (inclusive) OR operator
^	Bitwise exclusive OR operator

Questions

- Use command line arguments
- Flow controls
  - Selection & iterations
- Classes and objects
- Objects and reference variables
- JVM stack and garbage-collection heap
- Primitive types and variables

About W01-2_01-31_0

- Simple & compound statement?

Assignments

- CodeLab assignments