COMP 110/L Lecture 10

Mahdi Ebrahimi

Some slides adapted from Dr. Kyle Dewey

Outline

- Java Identifier
- “Random” numbers
- `if / else if /... /else`

Java Identifier

- In programming languages, identifiers are used for identification purpose.
- In Java, an identifier can be a *class name*, *method name*, or a *variable name*.

Java Identifier Example

```
public class Test
{
    public static void main(String[] args)
    {
        int num = 20;
    }
}
```

In the above java code, we have 5 identifiers namely:

Test : class name.
main : method name.
String : predefined class name.
args : variable name.
num : variable name.

Rules for defining Java Identifiers

There are certain rules for defining a valid java identifier. These rules must be followed, otherwise we get compile-time error.

1. The only allowed characters for identifiers are all alphanumeric characters([**A-Z**],[**a-z**],[**0-9**]), ‘**$**‘(dollar sign) and ‘_‘ (underscore). For example “num@” is not a valid java identifier as it contain ‘@’ special character.
2. Identifiers should **not** start with digits(**[0-9]**). For example “123num” is a not a valid java identifier.
3. Java identifiers are **case-sensitive**.
4. There is no limit on the length of the identifier but it is advisable to use an optimum length of 4 – 15 letters only.
5. **Reserved Words** can’t be used as an identifier. For example “int if = 20;” is an invalid statement as if is a reserved word. There are **53** reserved words in Java.

Naming rules for identifiers

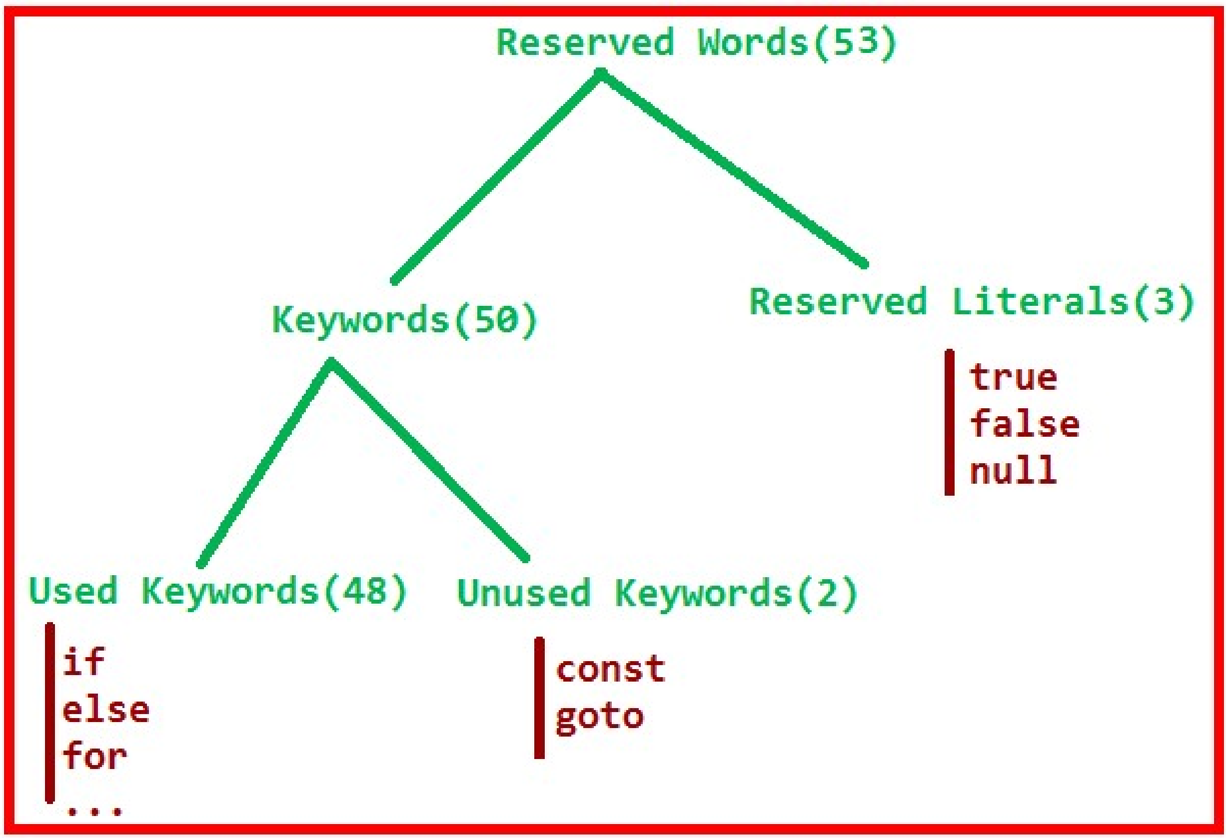
- Variable names are Case-sensitive in Java.
- Variable name can have Letter, digits and two special characters underscore and '$' sign.
- Name should with alphabet at the start , cannot start with number. Can use underscore and '$' sign.
- Following characters can be letters, digits, $ or _ character.
- White space , special characters like , * ; are not allowed.
- Variable name must not be a keyword (reserved word).

Example

Valid Identifier	Invalid Identifier
MyVariable	**My Variable** // contains a space
MYVARIABLE	**123num** // begins with a digit
myvariable	**a+c** // plus sign is not an alphanumeric character
x	**variable-2** // hyphen is not an alphanumeric character
i	**sum_&_difference** // ampersand is not an alphanumeric character
x1	
i1	
_myvariable	
$myvariable	
sum_of_array	
num123	

Reserved Words

Any programming language reserves some words to represent functionalities defined by that language. These words are called reserved words. They can be briefly categorized into two parts : **keywords**(50) and **literals**(3).
keywords define functionalities and literals defines a value.



Java Keywords

abstract	assert	boolean	break	byte
case	catch	char	class	continue
default	do	double	else	enum
extends	final	finally	float	for
if	implements	import	instanceof	int
interface	long	native	new	package
private	protected	public	return	short
static	strictfp	super	switch	synchronized
this	throw	throws	transient	try
void	volatile	while		

Keywords that are not currently used

const	goto

Random Numbers

Random Numbers

Random numbers can be generated with

`java.util.Random`

Random Numbers

Random numbers can be generated with `java.util.Random`

```
Random r = new Random();
int isRandom = r.nextInt();
```

(generates any random integer)

Random Numbers

Random numbers can be generated with
`java.util.Random`

```
Random r = new Random();
int isRandom = r.nextInt();
```

(generates any random integer)

```
Random r = new Random();
int isRandom = r.nextInt(10);
```

Random Numbers

Random numbers can be generated with
`java.util.Random`

```
Random r = new Random();
int isRandom = r.nextInt();
```

(generates any random integer)

```
Random r = new Random();
int isRandom = r.nextInt(10);
```

(generates one of the following random integers:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

Example:

`RandomExample.java`

How `Random` Works

https://www.youtube.com/watch?v=aSlkVy3mbR0

How `Random` Works

- Not actually random, but *psuedorandom*
- General idea:
  - Start with a *seed* value
  - Do a computation on it
  - Computation produces a psuedorandom value and a new seed
  - Repeat for infinity

Passing Seed Values

Seeds can be explicitly passed to `Random`

Passing Seed Values

Seeds can be explicitly passed to `Random`

```
Random r = new Random(123);
// seed is 123
int isRandom = r.nextInt();
```

Passing Seed Values

Seeds can be explicitly passed to `Random`

```
Random r = new Random(123);
// seed is 123
int isRandom = r.nextInt();
```

Always produces `-1188957731`

Example:
`RandomExampleWithSeed.java`

Utility of Setting Seeds

Predictable random values mean predictable tests.

Utility of Setting Seeds

Predictable random values mean predictable tests.

```
@Test
public void testRandomCalculation() {
  long seed = 1231;
  assertEquals(Something.calc(seed),
               42);
}
```

Without Explicit Seeds

If no seed is passed, `Random` will generate a seed based off of another source, such as the current time.

Without Explicit Seeds

If no seed is passed, `Random` will generate a seed based off of another source, such as the current time.

```
Random r = new Random();
int isRandom = r.nextInt();
```

Random number between min and max

1- Using Math.random():

```
(int)(Math.random() * ((max - min) + 1)) + min
```

2- Using util.Random()

```
Random r = new Random();
int rand = r.nextInt((max - min) + 1) + min;
```

`if / else if /... /else`

`if` / `else`

So far: only two branches allowed

`if/else`

So far: only two branches allowed

```
if (x > 5) {
  return 7;
} else {
  return 8;
}
```

`if` / `else` With More Than Two Branches

More branches are possible

`if` / `else` With More Than Two Branches

More branches are possible

```
if (x == 0) {
  return 7;
} else if (x < 10) {
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Example:

`IfElseIfElse.java`

Note on Testing

Good idea to have at least one test for each branch

Note on Testing

Good idea to have at least one test for each branch

```
if (x == 0) {
  return 7;
} else if (x < 10) {
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {
  return 7;
} else if (x < 10) {
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {   0
  return 7;
} else if (x < 10) {
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {   0
  return 7;
} else if (x < 10) { 1
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {   0
  return 7;
} else if (x < 10) { 1
  return 8;
} else if (x > 50) { 51
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {   0
  return 7;
} else if (x < 10) { 1
  return 8;
} else if (x > 50) { 51
  return 9;
} else {   50
  return 10;
}
```

Example:
`IfElseIfElseTest.java`