Core Java

4. Primitives, Casts, Operators

Primitives

- Primitives are types of what can actually be processed by the processors
- Primitives have fixed properties
- In a pure OO language, there are no primitives. But why primitives in Java?
- Primitives are building blocks of Objects in Java
- Are Strings primitives?

Disadvantages of Primitives

- Simple concepts like Strings can not be primitives
- Primitives aren't very polymorphic
- Primitives aren't useful in collections of objects
- Primitives require boilerplate code for serializing
- Solution: Autoboxing

List of Primitives

boolean (1B) true,false

byte (1B)

short (2B) / char (2B)

int (4B)

long (8B)

float (4B) IEEE 32

double (8B) IEEE 64

Character Literals

- `'character'`
- `'character'` cannot be a `'` or a `\`
- Escape Sequences otherwise
- Supported: `\b \t \n \f \r \” \' \\`
- Octal: `\u0000-00ff`

Operators

- Comparison Operators result in `boolean`
- Numeric Operators
- Cast Operator

Comparison Operators

== !=

< <= >= >

!

&&

||

Numeric Operators

```
+ - (Unary)
+ - * /
% (Integral types)
++ -- (Pre/Post)
& | ^ (Integral types)
~ (Integral types)
<< >> >>> (Integral types)
```

Cast Operator

```
type1 t1;
...
type2 t2=(type2)t1;
```

Casts

- Casts are applicable for numeric types
- Widening casts are implicit
- Loss of precision due to widening casts will not result in compilation errors
- Narrowing casts (even within same class) must be explicit
- Casts are not applicable to *boolean*s

Demonstration

- Compile and Execute a few programs

Questions?