CSE310: Programming in Java

Fundamentals of Programming in Java

Naming Conventions

- Use lowercase for variables and methods. If a name consists of several words, concatenate them into one, making the first word lowercase and capitalizing the first letter of each subsequent word—for example, the variables radius and area and the method print.
- Capitalize the first letter of each word in a class name—for example, the class names : ComputeArea and System.
- Capitalize every letter in a constant, and use underscores between words—for example, the constants PI and MAX_VALUE.

It is important to follow the naming conventions to make your programs easy to read.

Identifiers

- A name in a program is called an identifier.
- Identifiers can be used to denote classes, methods, variables, and labels.
- An identifier may be any descriptive sequence of uppercase and lowercase letters, numbers, or the underscore and dollar-sign characters.
- An identifier must start with a letter, an underscore (**_**), or a dollar sign (**$**). It cannot start with a digit.

Example: number, Number, sum_$, bingo, $$_100

Keywords

Keywords are reserved identifiers that are predefined in the language.

Cannot be used as names for a variable, class, or method.

All the keywords are in lowercase.

There are 50 keywords currently defined in the Java language.

The keywords **const** and **goto** are reserved but not used.

true, false, and **null** are also reserved.

Java Keywords

The following fifty keywords are reserved for use by the Java language:

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

The keywords `goto` and `const` are C++ keywords reserved, but not currently used in Java. This enables Java compilers to identify them and to produce better error messages if they appear in Java programs.

The literal values `true`, `false`, and `null` are not keywords, just like literal value `100`. However, you cannot use them as identifiers, just as you cannot use `100` as an identifier.

In the code listing, we use the keyword color for `true`, `false`, and `null` to be consistent with their coloring in Java IDEs.

Numeric literals

A literal is a constant value that appears directly in a program.

Integer literals(2,98,-45 etc.)

Floating point literals(2.34f,4.675d)

Scientific notations(1.2345e2,1.5678e-2)

Other literals

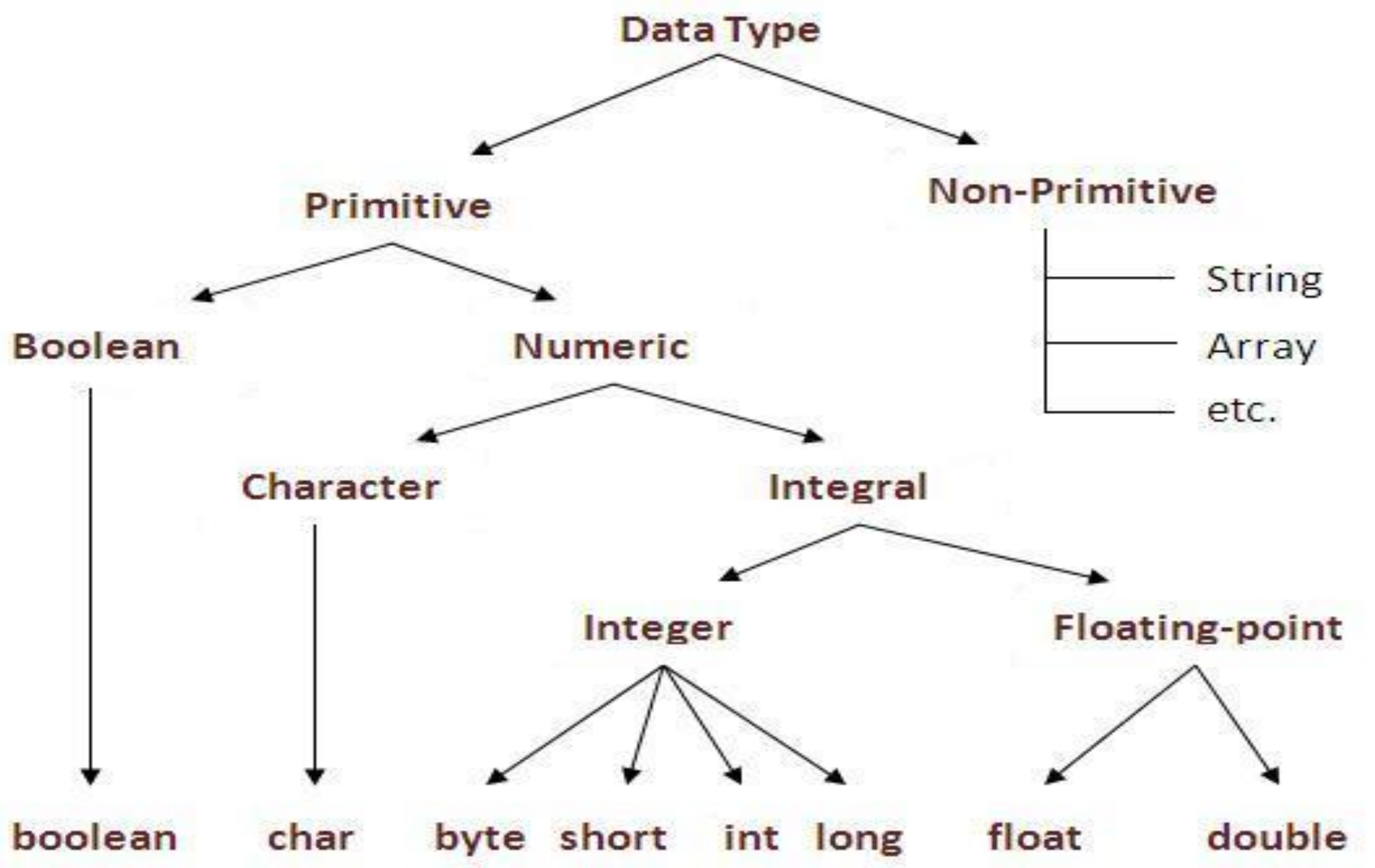
Boolean literals: true and false

Character literals: Always enclosed in single quotes, e.g. ‘A’, ’@’

String literals: Always enclosed in double quotes

““,e.g. “Programming”, “Hello”

Data types in Java



CODERINDEED

Storage size and default values

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

CODERINDEED

Ranges

Integers:

Name	Width	Range
long	64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	–2,147,483,648 to 2,147,483,647
short	16	–32,768 to 32,767
byte	8	–128 to 127

Floating-point types:

Name	Width in Bits	Approximate Range
double	64	4.9e–324 to 1.8e+308
float	32	1.4e–045 to 3.4e+038

Ranges

Character:

- Java uses Unicode to represent characters
- Unicode defines a fully international character set that can represent all of the characters found in all human languages.
- At the time of Java's creation, Unicode required 16 bits. Thus, in Java char is a 16-bit type. The range of a char is 0 to 65,536.

Type conversion and Casting

- ➢ It is fairly common to assign a value of one type to a variable of another type. If the two types are compatible, then Java will perform the conversion automatically.
- ➢ When one type of data is assigned to another type of variable, an automatic type conversion will take place if the following two conditions are met:

- *The two types are compatible.*
- *The destination type is larger than the source type.*

- ➢ When these two conditions are met, a **widening** conversion takes place. For example, the **int** type is always large enough to hold all valid **byte** values, so no explicit cast statement is required.
- ➢ if we try to assign int typed value(larger ranged value) to byte(smaller ranged value), error would arise.[As here we are doing wrong assignment, so java compiler will not perform narrowing automatically]
- ➢ If we want to perform narrowing, we need to use explicit type casting

More points

In java the numeric data types are compatible with each other but no automatic conversion is supported from numeric type to char or boolean. Also, char and boolean are not compatible with each other.

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

Examples: Type conversion and casting

<u>Example 1:</u>

byte x=123;

int y=x;[Widening-Type conversion]//No error

<u>Example 2:</u>

int y=123;

byte x=y;[Error-Incompatible types]

<u>Example 3:</u>

int y=123;

byte x=(int)y;[Narrowing-Using cast expression]//No error

Automatic type promotions in java

Java defines several *type promotion* rules that apply to expressions. They are as follows:

- First, all **byte**, **short**, and **char** values are promoted to **int**, as just described.
- Then, if one operand is a **long**, the whole expression is promoted to **long**.
- If one operand is a **float**, the entire expression is promoted to **float**.
- If any of the operands are **double**, the result is **double**.

Example-Type promotion

```
public class Promote {
public static void main(String args[]) {
byte b = 42;
char c = 'a';
short s = 1024;
int i = 50000;
float f = 5.67f;
double d = .1234;
double result = (f * b) + (i / c) - (d * s);
System.out.println((f * b) + " + " + (i / c) + " - " + (d * s));
}
```

In the first subexpression, **f * b, b** is promoted to a **float** and the result of the subexpression is **float**. Next, in the subexpression **i/c, c** is promoted to **int**, and the result is of type **int**. Then, in **d * s**, the value of **s** is promoted to **double**, and the type of the subexpression is **double**. Finally, these three intermediate values, **float**, **int**, and **double**, are considered. The outcome of **float** plus an **int** is a **float**. Then the resultant **float** minus the last **double** is promoted to **double**, which is the type for the final result of the expression.

Writing Your First Java Program

- class MyJavaProgram
-     {
-             public static void main(String args[])
                  {
                          System.out.println(“Have fun in Java…”);
                  }
-     }

- Understanding first java program
- Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().
- **class** keyword is used to declare a class in java.
- **public** keyword is an access modifier which represents visibility, it means it is visible to all.
- **static** is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.
- **void** is the return type of the method, it means it doesn't return any value.
- **main** represents startup of the program.
- **String[] args** is used for command line argument.
- **System.out.println()** is used print statement.
-

- System.out.println is a Java statement that prints the argument passed, into the **System.out** which is generally stdout.
- System is a Class
- out is a Static Member Field
- println() is a method
- System is a class in the **java.lang package** . The out is a static member of the System class, and is an instance of **java.io.PrintStream** . The println is a method of java.io.PrintStream. This method is overloaded to print message to output destination, which is typically a console or file.
- class System {
- public static final PrintStream out;
- //...
- }

  the System class belongs to java.lang package

```
class PrintStream{
  public void println();
  //...
}
```

  the Prinstream class belongs to java.io package

-

Compiling and Executing Java Program

Step-1: Save your file with .java extension.

- Example: Program1.java

NOTE: If the class is public then the file name MUST BE same as the name of the class.

Step-2: Compile your .Java file using javac compiler from the location where the file is saved.

```
javac Program1.java
```

Compiling and Executing Java Program

Step-3: Execute your java class which contains the following method: public static void main(String args[]) {}

```
java MyJavaProgram
```

Q1

Which of the following is an invalid identifier?

A. $abc

B. _abc

C. abc_pqr

D. #abc

Q2

Which of the following is valid identifier name?

A. abc#pqr

B. 1abc

C. a$b

D. @abc

Q3

What will be the output of following code?

```
public class abc2
{
public static void main(String[] args)
{
long x=123;
int y=x;
System.out.println(y);
}
}
```

A. 123

B. 123000

[✓ C.] Compile time error

D. 0

Q4

What will be the output of following code?

```
public class abc2
{
public static void main(String[] args)
{
double x=123.56;
int y=(int)x;
System.out.println(y);
}
}
```

A. 123.0

[✓ B.] 123

C. Compile time error

D. Runtime error

Q5

What will be the output of following code?

```
public class abc2
{
public static void main(String[] args)
{
byte a=127;
a++;
System.out.println(a);
}
}
```

A. 128

[✓ B.] -128

C. 0

D. Compile time error

Q6

Output?

```
public class abc2
{
public static void main(String[] args)
{
char x=65;
System.out.println(x);
}
}
```

A. A

B. 65

[✓ C.] Garbage value

D. Some special character will be printed

What will be the output of following code?(Q7)

```
public class First
{
public static void main(String[] args)
{
int a=0xC;
int b=0b1111;
System.out.println(a+" "+b);
}
}
```

A. 12 14

B. 11 10

C. Error

[✓ D. 12 15]

Q8

The smallest integer type is.........and its size is.......bits.

A.short, 8

B.byte, 8

C.short, 16

D.short, 16

CODERINDEED

Q9

In Java,the word *true* is

A. A Java keyword

B. A Boolean literal

[✓ C. Same as value 1]

D. Same as value 0

Q10

Automatic type conversion in Java takes place when

A. Two type are compatible and size of destination type is shorter than source type.

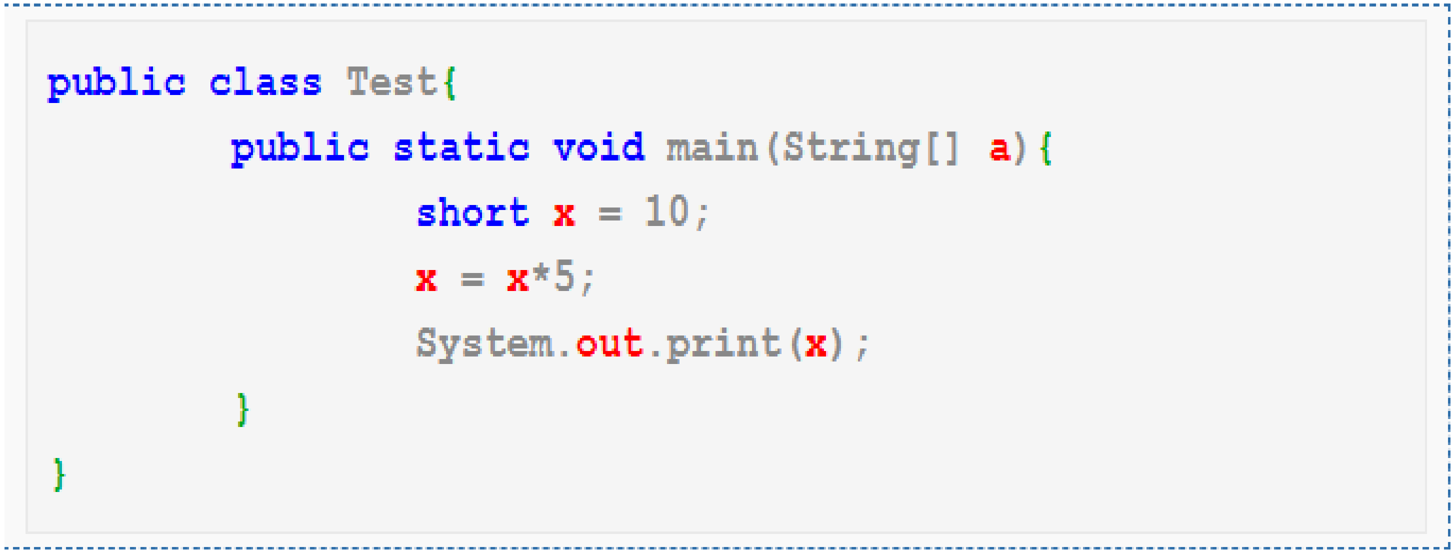
B. Two type are incompatible and size of destination type is equal of source type.

C. Two type are compatible and size of destination type is larger than source type.

D. All of the above

CODERINDEED

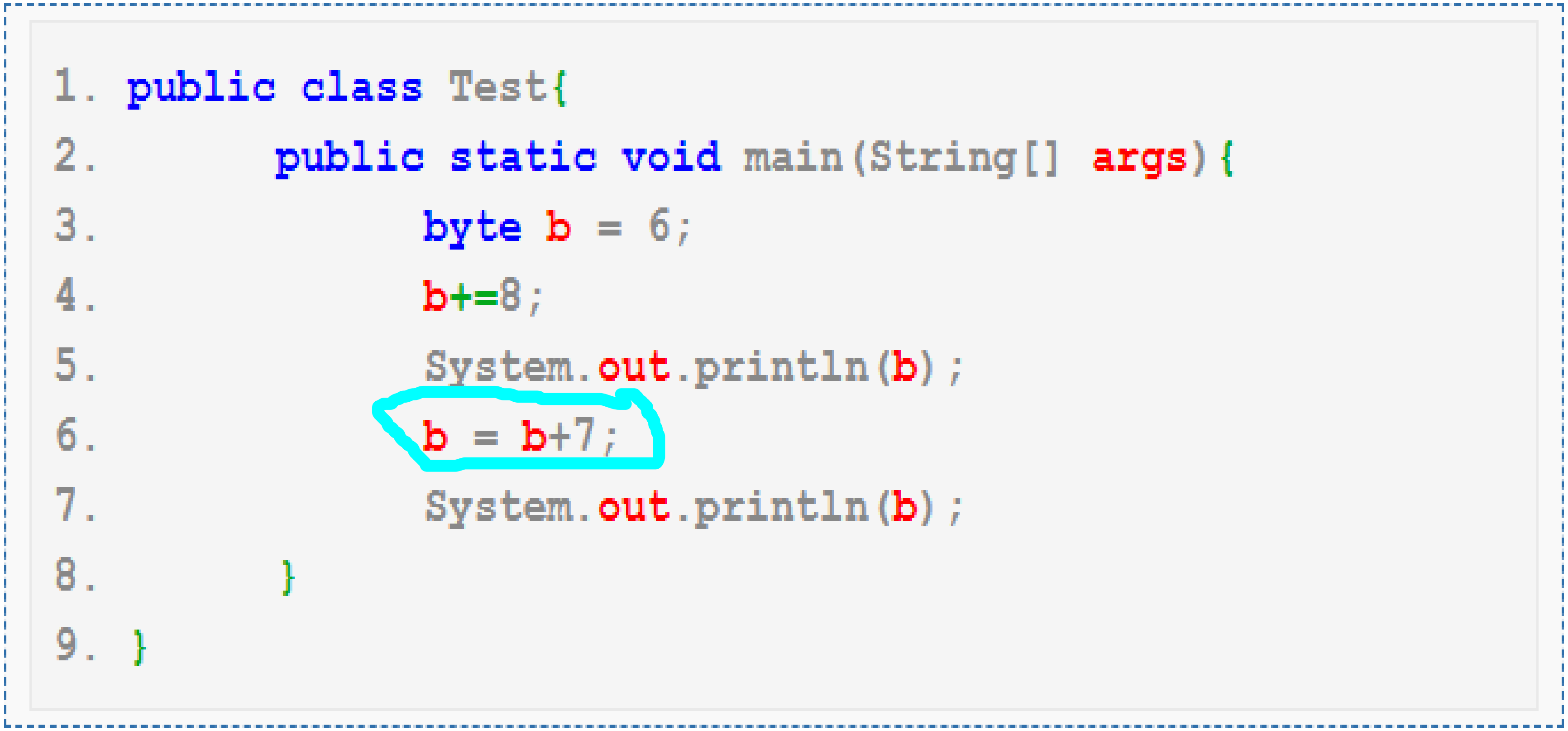
Q11



```
public class Test{
        public static void main(String[] a){
                short x = 10;
                x = x*5;
                System.out.print(x);
        }
}
```

A.50
B.10
[✓ C.Compilation Error]
D.None of these

Q12



```
1. public class Test{
2.         public static void main(String[] args){
3.                 byte b = 6;
4.                 b+=8;
5.                 System.out.println(b);
6.                 b = b+7;
7.                 System.out.println(b);
8.         }
9. }
```

A. 14 21
B. 14 13
C. Compilation fails with an error at line 6
D. Compilation fails with an error at line 4