Java Basic

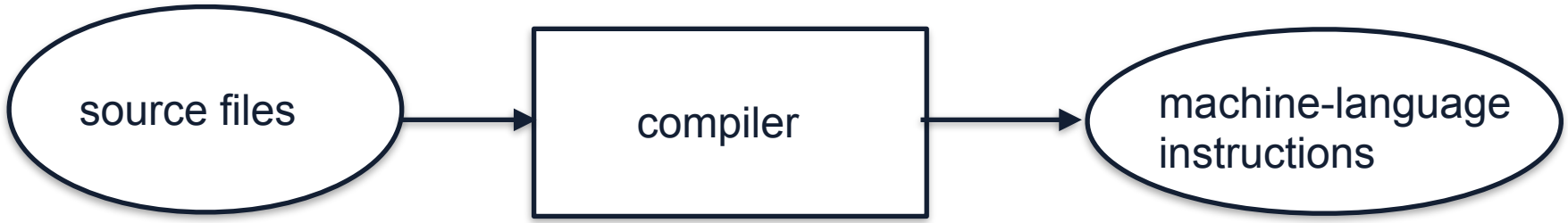
CS 284 C
Instructor: Susan Liu
xueqing.liu@stevens.edu

Learning Objectives

- Java basic:
  - Java environment (JVM) and classes
  - Primitive data types and reference variables
  - the Math class
  - String class
  - Wrapper class for primitive types
  - Defining your own class
  - Array
  - Java I/O

Java Virtual Machine (JVM)

- Introduced in 1995 by Sun company
- Write once, run anywhere (WORA)

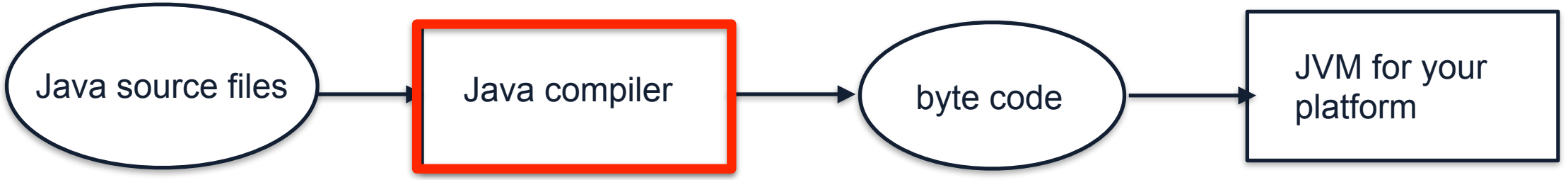


platform dependent, e.g., assembly

(traditional way)

Java Virtual Machine (JVM)

- Introduced in 1995 by Sun company
- Write once, run anywhere (WORA)



(platform **independent**)

(platform **dependent**, e.g., Windows, Linux, Mac)

A: dependent

Java Classes

- A class is a description of a group of entities (objects) that share the same characteristics

```
public class Person {
  // Data Fields
  /** The given name */
  private String givenName = "Mary";
  /** The age*/
  private int age = 30;
}
```

person 1: Mary, age = 30
person 2: Susan, age = 53
…

class

objects

Java Method

- A method is a collection of statements that provide some tasks and return the result

```
public class Person {
  /** getting the age of a person */
  public int getAge(int birthYear){
      return 2020 - birthYear;
  }
}
```

```
age = getAge(1990);
System.out.println(age);
```

Output: 30

Data Fields and Types

- Variables must be declared with a type before use (unlike Python)

```
private String givenName = “Mary”; // Java
```

```
givenName = “Mary”; #Python
```

- Primitive types (numbers, characters) vs. objects types
- 8 primitive types

byte	-128 to 127
short	-32,768 to 32,767
int	-2,147,483,648 to 2,147,483,647
long	-2^{63} to $2^{63}-1$
float	32-bit IEEE 754 floating point
double	64-bit IEEE 754 floating point
char	Unicode character set
boolean	true, false

Type Compatibility and Conversion

- Widening conversion:
  - int -> double 
  - double -> int 

```
int item = 42;
double realItem = item; // valid

double y = 3.14;
int x = y;
“Compile-time Error: Type mismatch: cannot
convert from double to int”
```

Java Constructor Method

- The constructor method initializes the values of an object

```
public class Person {
  public Person(String givenName, String ID, int age)
{
    …..
}
  public Person(int age){
    …..
}
}
```


Constructor methods have no return type

```
Person mary = new Person("Mary", '123', 23);

Person susan = new Person("Susan", '456', 53);

Person susan = new Person(23);

Person susan = new Person();
```

The `main` Method

- The point where execution begins

```
public class Person {
  public Person(String givenName, String ID, int age) {
    …..
}
  public static void main(String[] args){
     Person mary = new Person(“Mary”, ‘123’, 23);
     …..
}
}
```

Modifying/Getting Values of Objects

- Use the set and get method to modify/get the values of an object

```
public class Person {
  private int age;
  public void setAge(int age) {
      this.age = age;
}
  public String getAge(){
      return this.age;
}
}
```

this refers to the current object

```
 public static void main(String[] args){
Person mary = Person();
mary.setAge(23);
System.out.println(mary.getAge());
}
```

```
 public static void main(String[] args){
Person mary = Person();
mary.age = 23;  ??
System.out.println(mary.age); ??
}
```

Testing Java Methods

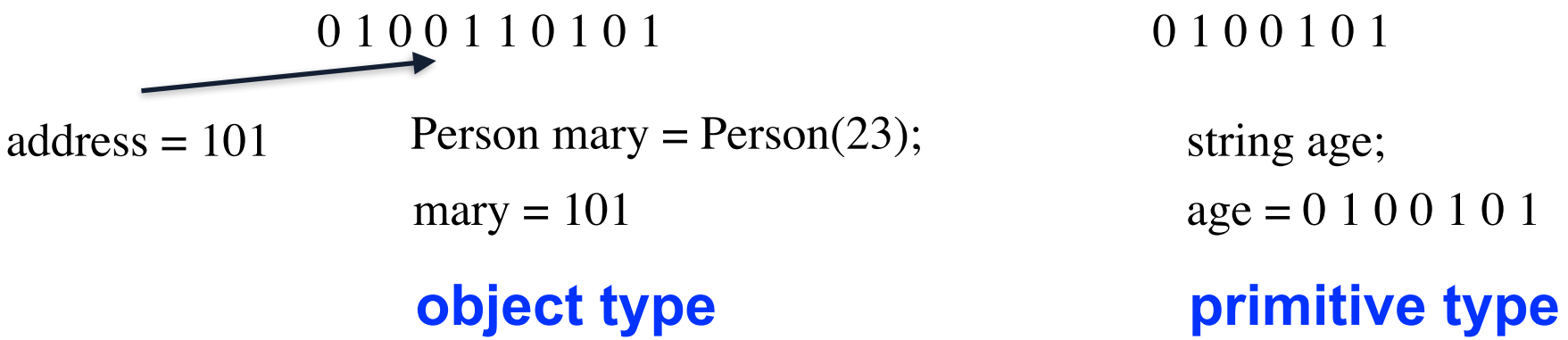
```
public class TestPerson {
 public static void main(String[] args) {
   Person mary = new Person("Mary","123", 30);
   Person susan = new Person("Susan", "456", 53);

   System.out.println("Age of  Mary is " + mary.getAge());
   // prints: Age of Mary is 30

   mary.setAge(35);

   System.out.println("Age of  Mary is " + mary.getAge());
  // prints: Age of Mary is 35
}
```

Referencing Objects



- The Person object Mary is now referenced by the variable `mary`
- `mary` stores the address in memory where the specific object Mary is stored
- Primitive types store the **values** instead of addresses
- Demo 1: Person.java

Static Variable

```
public static int age_static = 30;
```

- Static variables are class variables
  - Shared across all instances
  - Allocated only 1 time

- Instance variables
  - Belong to a specific object
  - Allocated once every object is created

- Demo: Person_2.java

Static Method

- Methods that can be called before any objects being constructed

```
public class Car {
  public void setMileage(int mileage) {
      this.age = age;
}
  public static void convertMpgToKpl(int Mpg){
      ……
}
}
```

The `Math` Class

- Collection of useful math operations
- All static

Method	Behavior
`static` *numeric* `abs(`*numeric*`)`	Returns the absolute value of its *numeric* argument (the result type is the same as the argument type)
`static double ceil(double)`	Returns the smallest whole number that is not less than its argument
`static double cos(double)`	Returns the trigonometric cosine of its argument (an angle in radians)
`static double exp(double)`	Returns the exponential number *e* (i.e., 2.718 . . .) raised to the power of its argument
`static double floor(double)`	Returns the largest whole number that is not greater than its argument
`static double log(double)`	Returns the natural logarithm of its argument

Static Variable

```
public static int age_static = 30;
```

- Static variables are class variables
  - Shared across all instances
  - Allocated only 1 time

- Instance variables
  - Belong to a specific object
  - Allocated once every object is created

- Demo: Person_2.java