CS 149

Professor: Alvin Chao

CS149 – More with Classes and Objects

JAMES MADISON UNIVERSITY®

OverLoading

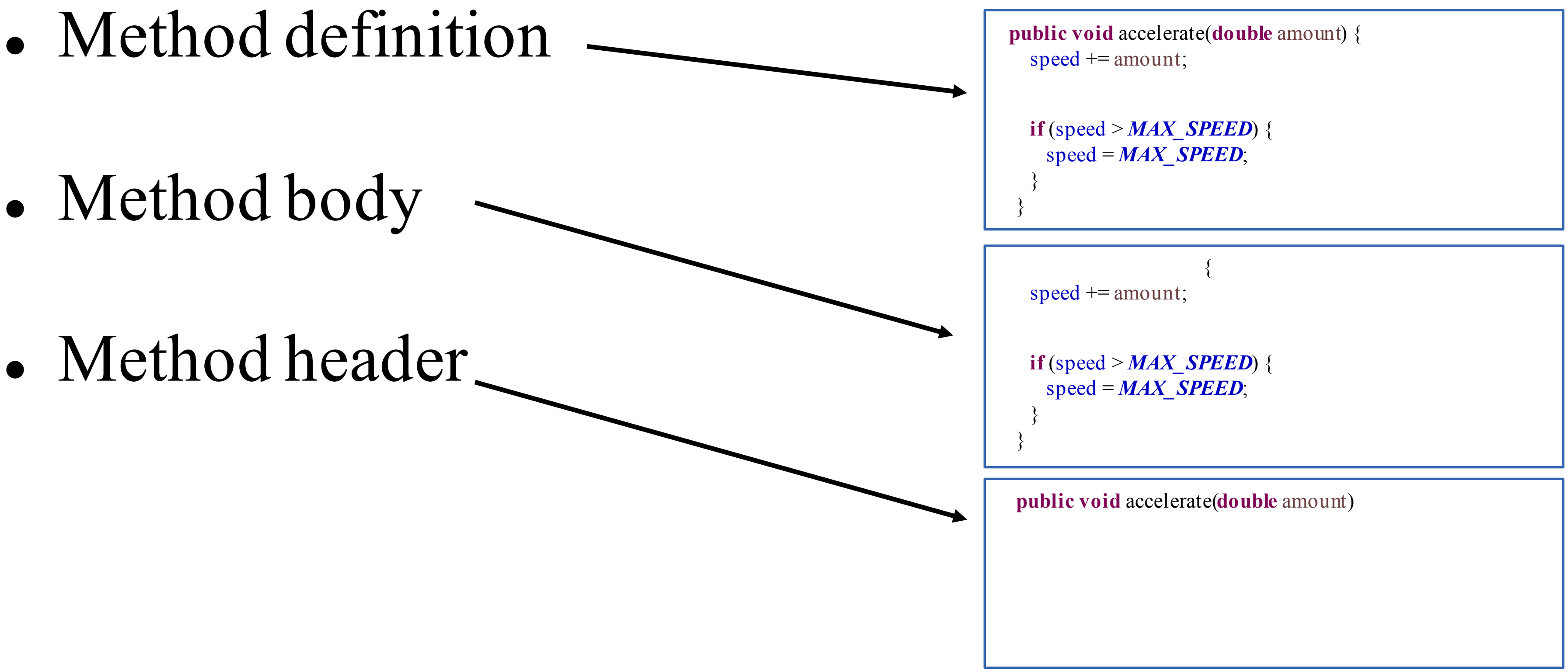
- Let's look at the Car class...

Terminology

- Method definition

```
public void accelerate(double amount) {
   speed += amount;


   if (speed > MAX_SPEED) {
      speed = MAX_SPEED;
   }
 }
```

Terminology



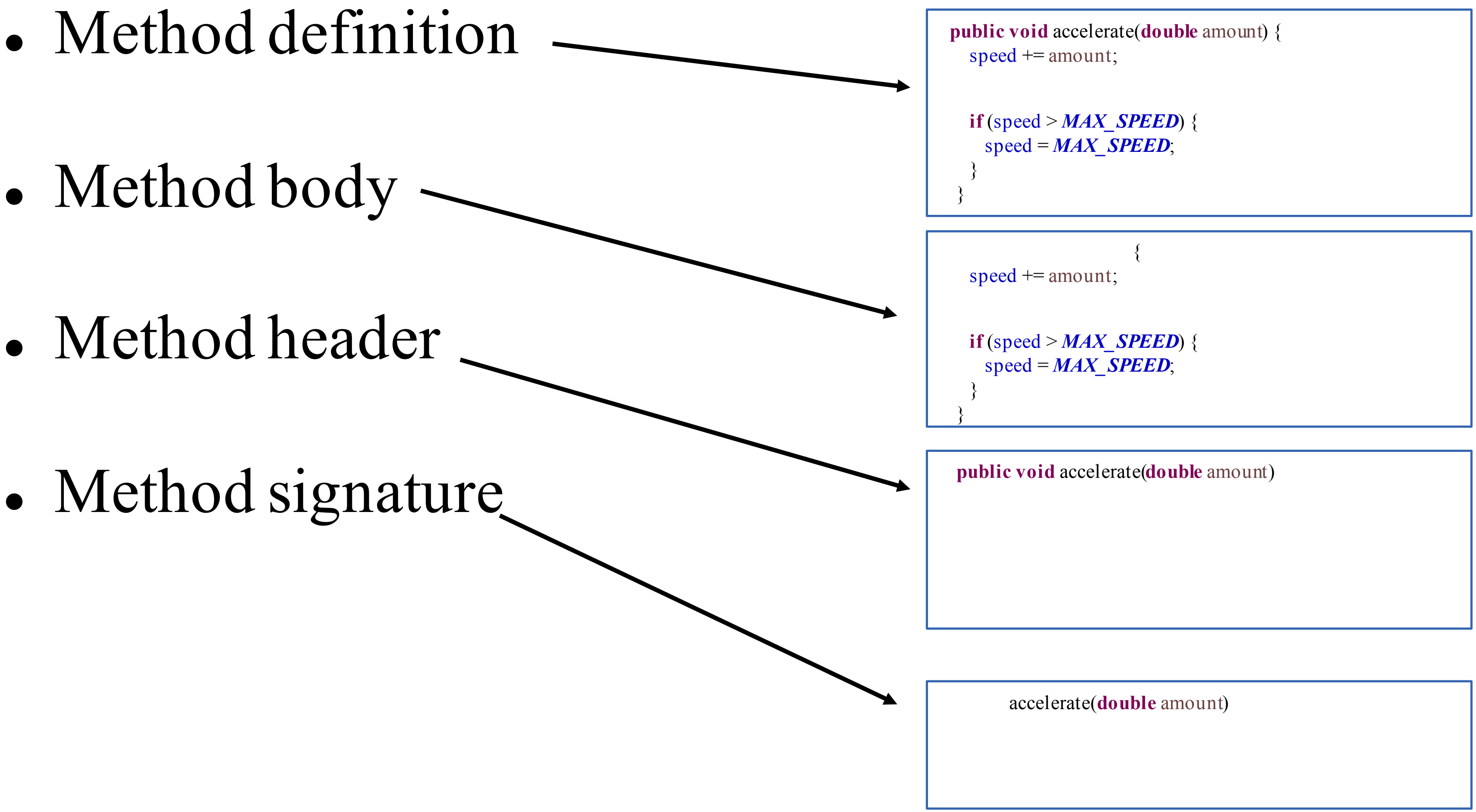
- Method definition
- Method body
- Method header

```
public void accelerate(double amount) {
  speed += amount;


  if (speed > MAX_SPEED) {
    speed = MAX_SPEED;
  }
}
```

```
                                     {
  speed += amount;


  if (speed > MAX_SPEED) {
    speed = MAX_SPEED;
  }
}
```

```
public void accelerate(double amount)
```

Terminology



- Method definition
- Method body
- Method header
- Method signature

```
public void accelerate(double amount) {
  speed += amount;


  if (speed > MAX_SPEED) {
    speed = MAX_SPEED;
  }
}
```

```
                                  {
  speed += amount;


  if (speed > MAX_SPEED) {
    speed = MAX_SPEED;
  }
}
```

```
public void accelerate(double amount)
```

```
accelerate(double amount)
```

Quiz #1

```
public class Person {

   private String name;

   public Person(String name) {
      name = name;
   }

   public String getName() {
      return name;
   }

}
```

```
Person bob = new Person("Bob");
System.out.println(bob.getName());
```

Will it compile?

If so, what will be printed?

Quiz #1

```
public class Person {

  private String name;

  public Person(String name) {
    name = name;
  }

  public String getName() {
    return name;
  }

}
```

```
Person bob = new Person("Bob");
System.out.println(bob.getName());
```

The name parameter shadows the name field.

Will it compile? YES

If so, what will be printed? null

Quiz #2

```
public class Person {

   private String name;

   public Person(String n) {
      name = n;
   }

   public String getName() {
      return name;
   }

}
```

```
Person bob = new Person("Bob");
System.out.println(bob.toString());
```

Will it compile?

If so, what will be printed?

Quiz #2

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

}
```

```
Person bob = new Person("Bob");
System.out.println(bob.toString());
```

Will it compile? YES

If so, what will be printed? Person@6ff4ff23

Quiz #2

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

}
```

```
Person bob = new Person("Bob");
System.out.println(bob.toString());
```

Every class has a default toString method. The string will contain the name of the class and the location of the object in memory. (This is usually not very helpful.)

Will it compile? YES

If so, what will be printed? Person@6ff4ff23

Quiz #3

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

 public String toString() {
    return "Person named: " + name;
  }

}
```

```
Person bob = new Person("Bob");
System.out.println(bob);
```

Will it compile?

If so, what will be printed?

Quiz #3

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

 public String toString() {
    return "Person named: " + name;
  }

}
```

```
Person bob = new Person("Bob");
System.out.println(bob);
```

Will it compile? YES

If so, what will be printed?

Person named: Bob

The print methods will automatically call the toString method of any object passed to them.

Quiz #4

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

 public String toString() {
    return "Person named: " + name;
  }

}
```

```
Person bob1 = new Person("Bob");
Person bob2 = new Person("Bob");
System.out.println(bob1 == bob2);
```

Will it compile?

If so, what will be printed?

Quiz #4

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

 public String toString() {
    return "Person named: " + name;
  }

}
```

```
Person bob1 = new Person("Bob");
Person bob2 = new Person("Bob");
System.out.println(bob1 == bob2);
```

Will it compile? YES

If so, what will be printed?

false

Quiz #4

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

 public String toString() {
    return "Person named: " + name;
  }

}
```

```
Person bob1 = new Person("Bob");
Person bob2 = new Person("Bob");
System.out.println(bob1 == bob2);
```

Will it compile? YES

If so, what will be printed?

false

== Compares the addresses (references) stored in the two variables.

Two objects → two different addresses.

Quiz #5

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

 public String toString() {
    return "Person named: " + name;
  }

}
```

```
Person bob1 = new Person("Bob");
Person bob2 = new Person("Bob");
System.out.println(bob1.equals(bob2));
```

Will it compile?

If so, what will be printed?

Quiz #5

```
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

 public String toString() {
    return "Person named: " + name;
  }

}
```

```
Person bob1 = new Person("Bob");
Person bob2 = new Person("Bob");
System.out.println(bob1 .equals(bob2));
```

Will it compile? YES

If so, what will be printed?

false

Quiz #5

```java
public class Person {

  private String name;

  public Person(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

  public String toString() {
    return "Person named: " + name;
  }

}
```

```java
Person bob1 = new Person("Bob");
Person bob2 = new Person("Bob");
System.out.println(bob1.equals(bob2));
```

Will it compile? YES

If so, what will be printed?

false

All classes get a default .equals method. It just uses ==.

Typical equals Method

```
public class Person {
  private String name;
  private int ssn;

  public Person(String nm, int num) {
    name = nm;
    ssn = num;
    }

  public String getName() {
    return name;
    }

  public int getSSN() {
     return ssn;
    }

  public String toString() {
    return "Person named: " + name;
     }
  public boolean equals(Person other) {

    return name.equals(other.getName())
         && ssn == other.getSSN();
     }
}
```

← Compares all fields.

Typical equals Method

```
public class Person {
  private String name;
  private int ssn;

  public Person(String nm, int num) {
    name = nm;
    ssn = num;
    }

  public String getName() {
    return name;
    }

  public int getSSN() {
     return ssn;
    }

  public String toString() {
    return "Person named: " + name;
     }
  public boolean equals(Person other) {


    return name.equals(other.getName())
          && ssn == other.getSSN();
     }
}
```

```
Person bob1 = new Person("Bob", 1);
Person bob2 = new Person("Bob", 1);
Person bob3 = new Person("Bob", 3);
Person bob4 = bob2;



System.out.println(bob1.equals(bob2));
System.out.println(bob1.equals(bob3));
System.out.println(bob1.equals(bob4));
System.out.println(bob2.equals(bob4));
System.out.println(bob1 == bob2);
System.out.println(bob1 == bob3);
System.out.println(bob1 == bob4);
System.out.println(bob2 == bob4);
```

What will be printed?

Typical equals Method

```
public class Person {
  private String name;
  private int ssn;

  public Person(String nm, int num) {
    name = nm;
    ssn = num;
    }

  public String getName() {
    return name;
    }

  public int getSSN() {
     return ssn;
    }

  public String toString() {
    return "Person named: " + name;
     }
  public boolean equals(Person other) {

    return name.equals(other.getName())
         && ssn == other.getSSN();
     }
}
```

```
Person bob1 = new Person("Bob", 1);
Person bob2 = new Person("Bob", 1);
Person bob3 = new Person("Bob", 3);
Person bob4 = bob2;


System.out.println(bob1.equals(bob2));
System.out.println(bob1.equals(bob3));
System.out.println(bob1.equals(bob4));
System.out.println(bob2.equals(bob4));
System.out.println(bob1 == bob2);
System.out.println(bob1 == bob3);
System.out.println(bob1 == bob4);
System.out.println(bob2 == bob4);
```

What will be printed?

```
true
false
true
true
false
false
false
true
```

toString and equals

- Providing toString and equals methods should be a routine part of developing a class.

- No toString method?
  - Complicates testing
- No equals method?
  - May lead to sneaky bugs
  - (Remember that a “broken” equals method will be provided by default.)

Quiz

```
public class CarMain {

  public static void main(String[] args) {
    int result;
    String s = "Hello";
    System.out.println(s);
    result = nonsense1(s);
    System.out.println(s);

    Car ford = new Car("Ford", 1992);
    System.out.println(ford.getSpeed());
    result = nonsense2(ford);
    System.out.println(ford.getSpeed());
    }
  public static int nonsense1(String word) {

    word += word;
    return word.length();
    }

  public static int nonsense2(Car car) {

    car.accelerate();
    return car.getYear();
    }

}
```

What will be printed?

Quiz

```
public class CarMain {

  public static void main(String[] args) {
    int result;
    String s = "Hello";
    System.out.println(s);
    result = nonsense1(s);
    System.out.println(s);

    Car ford = new Car("Ford", 1992);
    System.out.println(ford.getSpeed());
    result = nonsense2(ford);
    System.out.println(ford.getSpeed());
    }
  public static int nonsense1(String word) {

    word += word;
    return word.length();
    }

  public static int nonsense2(Car car) {

    car.accelerate();
    return car.getYear();
    }

}
```

What will be printed?

```
Hello
Hello
0.0
5.0
```

Mutable vs Immutable Types

- Immutable types – Objects can't be changed once created
  - String is an immutable type
- Mutable types – An object's fields may change over time

- We need to be careful that our methods don't accidentally modify mutable objects that are passed as arguments.

- **Acknowledgements**

Parts of this activity are based on materials developed by Chris Mayfield and Nathan Sprague.

</end>