ITP20003 Java Programming

Defining Classes
(Chapter 5)

This slide is primary taken from the instructor's resource of Java: Introduction to Problem Solving and Programming, 7th ed. by Savitch and then edited partly by Shin Hong

Objects and Classes (1/2)

- An object is a program construction which aggregates items and operations on a certain thing
  - members: items and operations
    - fields (items): primitive variable or other object
    - methods (operations): operations on the contained items according to the values of the contained items and given parameters
  - a member has attributes
    - e.g., accessibility (`public` or `private`), static, final

- A Class is a type of a certain kind of objects
  - declare members (names and their types)
  - define default values of the members

Objects and Classes (2/2)

- Java intends programmers to define objects and represent the purposed computation as sequences of interactions among objects

- This way of programming, Object-oriented programming, is known to be good for constructing and maintaining large and complex SW
  - locate related things closely (i.e., modular design)
  - use a consistent name for members having the same purpose
  - define new objects by reusing exiting ones

- Java supports many features in defining Java classes such that programmers can clearly and concisely represent abstractions on the target domain and sub-module designs

Class and Method Definitions

- Figure 5.1 A class as a blueprint

Class Name: `Automobile`

Data:

```
amount of fuel_________
speed _________
license plate _________
```

Methods (actions):

`accelerate:` **increase power output**

How: Press on gas pedal.

`decelerate:` **reduce speed**

How: Press on brake pedal.

Class and Method Definitions

- Figure 5.1 ctd.

First Instantiation:

Object name: `patsCar`

```
amount of fuel: 10 gallons
speed: 55 miles per hour
license plate: "135 XJK"
```

Second Instantiation:

Object name: `suesCar`

```
amount of fuel: 14 gallons
speed: 0 miles per hour
license plate: "SUES CAR"
```

Third Instantiation:

Object name: `ronsCar`

```
amount of fuel: 2 gallons
speed: 75 miles per hour
license plate: "351 WLF"
```

Objects that are instantiations of the class **`Automobile`**

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

Example

```java
/* Dog.java */
public class Dog
{
  public String name;
  public String breed;
  public int age;

  public void writeOutput() {
     System.out.println("Name: " + name);
     System.out.println("Breed: " + breed);
     System.out.println("Age in cal. years: " + age);
     System.out.println("Age in human years: " +
                        getAgeInHumanYears());
     System.out.println();
  }

  public int getAgeInHumanYears(){
    int humanYears = 0;
    if (age <= 2) {
      humanYears = age * 11;
    }
    else {
      humanYears = 22 + ((age-2) * 5);
    }
    return humanYears;
  }
}
```

```java
/* DogDemo.java */
public class DogDemo
{
  public static void main(String[] args)
  {
    Dog balto = new Dog();
    balto.name = "Balto";
    balto.age = 8;
    balto.breed = "Siberian Husky";
    balto.writeOutput();

    Dog scooby = new Dog();
    scooby.name = new String("Scooby”);
    scooby.age = 42;
    scooby.breed = new String("Great Dane”);

    System.out.println(scooby.name + " is a " +
                       scooby.breed + ".");
    System.out.print("He is " + scooby.age +
                       " years old, or ");
    int humanYears = scooby.getAgeInHumanYears();
    System.out.println(humanYears +
                       " in human years.");
  }
}
```

Methods

- When you use a method you "invoke" or "call" it

- Two kinds of Java methods
  - Return a single item
  - Perform some other action – a `void` method

- The method `main` is a `void` method
  - Invoked by the system
  - Not by the application program

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch

Methods

- Calling a method that returns a quantity
  - Use anywhere a value can be used

- Calling a void method
  - Write the invocation followed by a semicolon
  - Resulting statement performs the action defined by the method

Defining `void` Methods

- Most method definitions we will see as `public`
- Method does not return a value
  - Specified as a `void` method
- Heading includes parameters
- Body enclosed in braces `{ }`
- Think of method as defining an action to be taken

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

Methods That Return a Value

- Consider method **`getAgeInHumanYears( )`**

```
public int getAgeInHumanYears()
{
    int humanAge = 0;
    if (age <= 2)
    {
        humanAge = age * 11;
    }
    else
    {
        humanAge = 22 + ((age-2) * 5);
    }
        return humanAge;
}
```

- Heading declares type of value to be returned
- Last statement executed is **`return`**

Second Example – `Species` Class

- Class designed to hold records of endangered species
- `SpeciesFirstVersion.java`

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

The Keyword `this`

- Referring to instance variables outside the class – must use
  - Name of an object of the class
  - Followed by a dot
  - Name of instance variable
- Inside the class,
  - Use name of variable alone
  - The object (unnamed) is understood to be there

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

Local Variables

- Variables declared inside a method are called *local* variables
  - May be used only inside the method
  - All variables declared in method `main` are local to `main`
- Local variables having the same name and declared in different methods are different variables

Blocks

- Recall compound statements
  - Enclosed in braces { }
- When you declare a variable within a compound statement
  - The compound statement is called a *block*
  - The scope of the variable is from its declaration to the end of the block
- Variable declared outside the block usable both outside and inside the block

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

Parameters of Primitive Type

```
public int getPopulationIn10()
{
    int result = 0;
    double populationAmount = population;
    int count = 10;
```

- Recall method declaration in listing 5.3
  - Note it only works for 10 years
  - We can make it more versatile by giving the method a parameter to specify how many years

- **Ex. SpiecesSecondVersion.java**

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

Information Hiding

- Programmer using a class method need <u>not</u> know details of implementation
  - Only needs to know *what* the method does

- Information hiding:
  - Designing a method so it can be used without knowing details
  - Also referred to as *abstraction*

- Method design should separate *what* from *how*

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch

The `public` and `private` Modifiers

- Type specified as `public`
  - Any other class can directly access that object by name
- Classes generally specified as `public`
- Instance variables usually <u>not</u> `public`
  - Instead specify as `private`

- Ex. `SpeciesThirdVersion.java`

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch

Accessor and Mutator Methods

- When instance variables are private must provide methods to access values stored there
  - Typically named `getSomeValue`
  - Referred to as an accessor method
- Must also provide methods to change the values of the private instance variable
  - Typically named `setSomeValue`
  - Referred to as a mutator method

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

Constructor Method

- A constructor is a special method which is invoked at an object instantiation to initialize the field members of the new object

- Restriction
  - A constructor should have the same name as the Class name
  - A constructor must not have any `return` statement
  - …

- Ex. **`SpeciesFourthVersion.java`**

Encapsulation

- Consider example of driving a car
  - We see and use break pedal, accelerator pedal, steering wheel – know <u>what</u> they do
  - We do <u>not</u> see mechanical details of <u>how</u> they do their jobs
- Encapsulation divides class definition into
  - Class interface
  - Class implementation

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch

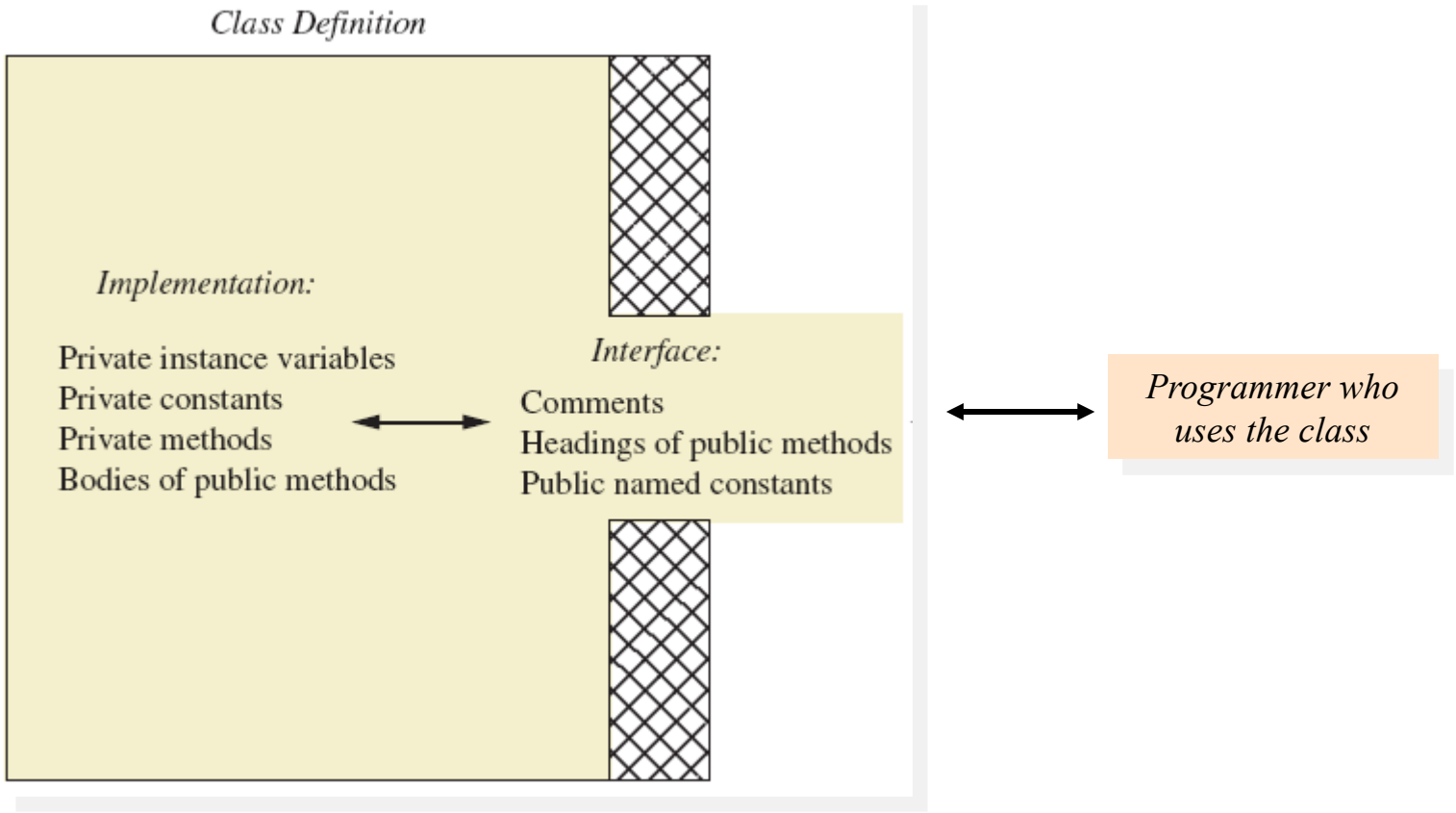
Encapsulation

- A *class interface*
  - Tells <u>what</u> the class does
  - Gives headings for public methods and comments about them
- A *class implementation*
  - Contains private variables
  - Includes definitions of public and private methods

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

Encapsulation

- Figure 5.3 A well encapsulated class definition



Encapsulation

- Preface class definition with comment on how to use class
- Declare all instance variables in the class as private.
- Provide public accessor methods to retrieve data Provide public methods manipulating data
  - Such methods could include public mutator methods.
- Place a comment before each public method heading that fully specifies how to use method.
- Make any helping methods private.
- Write comments within class definition to describe implementation details.

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch

Variables of a Class Type

- All variables are implemented as a memory location
- Data of *primitive type* stored in the memory location assigned to the variable
- Variable of *class type* contains memory address of object named by the variable

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

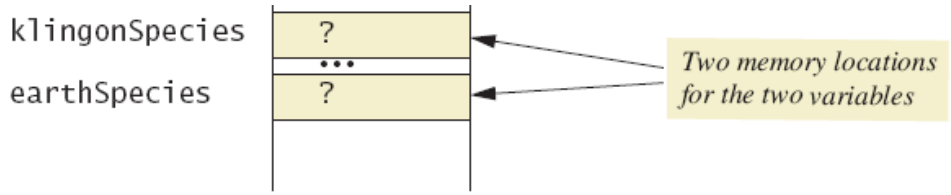
Variables of a Class Type

- Object itself not stored in the variable
  - Stored elsewhere in memory
  - Variable contains address of where it is stored
- Address called the *reference* to the variable
- A *reference type* variable holds references (memory addresses)
  - This makes memory management of class types more efficient

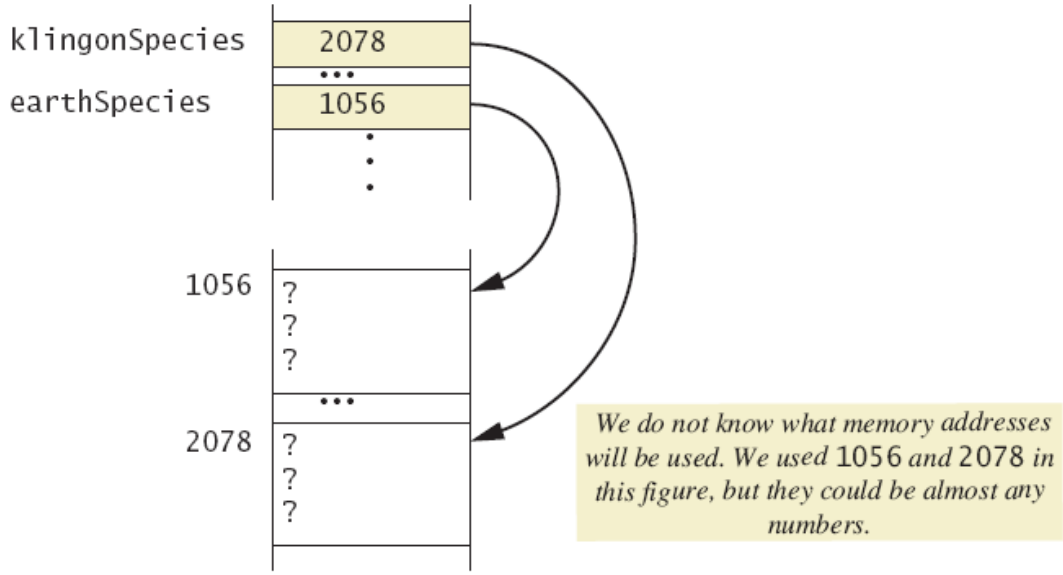
JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119

Variables of a Class Type

```
Species klingonSpecies, earthSpecies ;
```

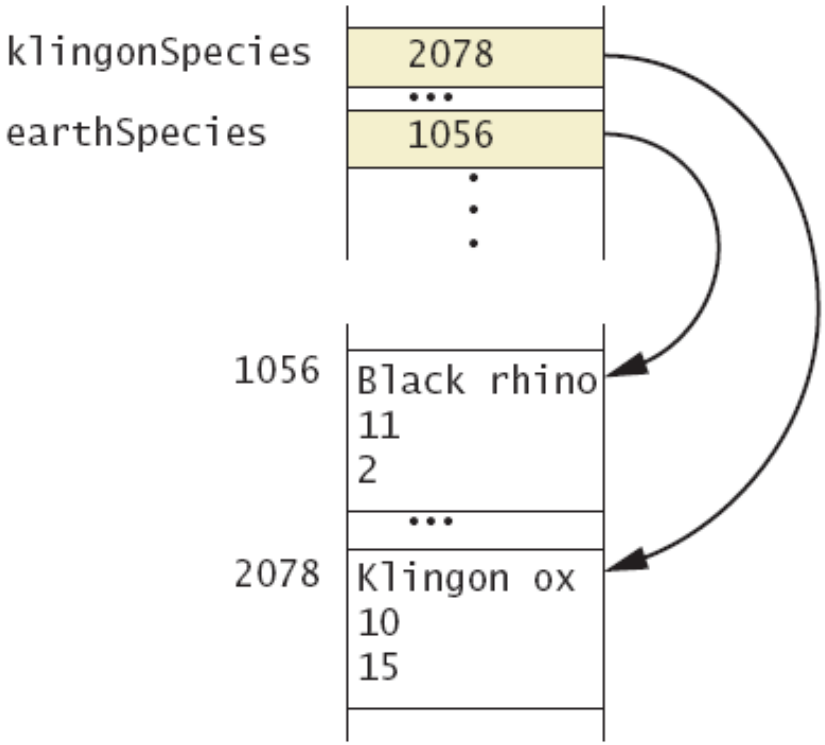



```
klingonSpecies = new Species() ;
earthSpecies = new Species() ;
```

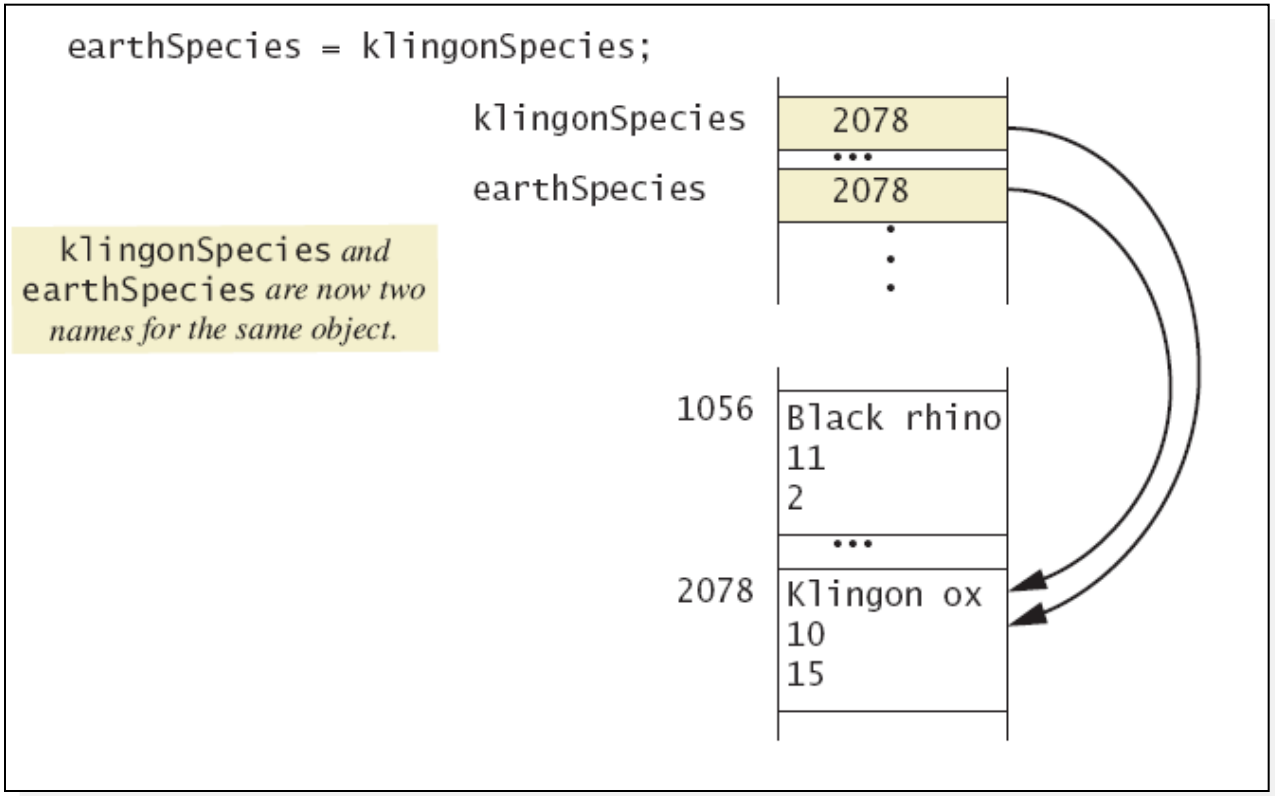


Variables of a Class Type

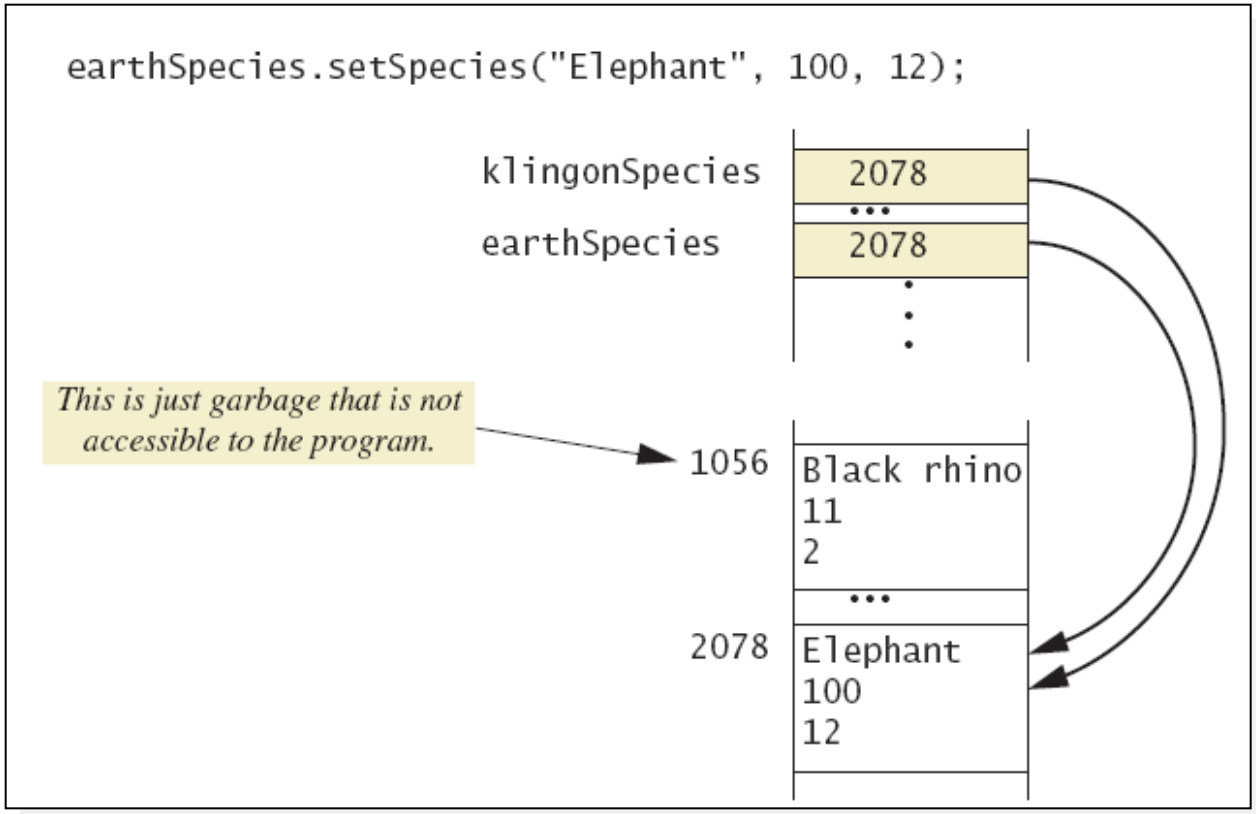
```
klingonSpecies.setSpecies("Klingon ox", 10, 15);
earthSpecies.setSpecies("Black rhino", 11, 2);
```



Variables of a Class Type

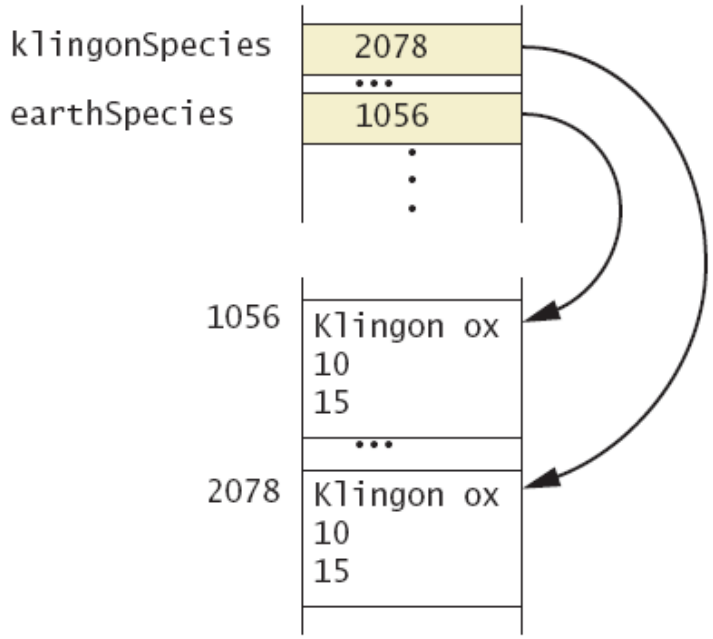


Variables of a Class Type



Variables of a Class Type

```
Species klingonSpecies = new Species() ;
Species earthSpecies = new Species() ;
klingonSpecies.setSpecies("Klingon ox", 10, 15);
earthSpecies.setSpecies("Klingon ox", 10, 15);
```




```
if (klingonSpecies == earthSpecies)
    System.out.println("They are EQUAL.");
else
    System.out.println("They are NOT equal.");
```

The output is `They are Not equal`, *because* 2078 *is not equal to* 1056.

Parameters of a Class Type

- When assignment operator used with objects of class type
  - Only memory address is copied
- Similar to use of parameter of class type
  - Memory address of actual parameter passed to formal parameter
  - Formal parameter may access public elements of the class
  - Actual parameter thus can be changed by class methods

JAVA: An Introduction to Problem Solving & Programming, 7th Ed. By Walter Savitch
ISBN 0133862119