CptS 355- Programming Language Design

Java Basics

Instructor: Sakire Arslan Ay

Washington State University
World Class. Face to Face.

Java Properties

- Strictly object oriented
- (Machine) architecture independent
- Type-Safety: No explicit pointers. All objects are accessed through implicit references.
- Robust
- Multi-threaded
- Has static scoping
- Has (mostly) static typing - otherwise dynamic strong typing
- Has garbage collection

“Hello, World!” in Java

`Greeter.java`

```
public class Greeter
{
        public Greeter(String aName) {
                name = aName;
        }
        public String sayHello(){
                return "Hello, " + name + "!";
        }
        private String name;
}
```

This class has three features:

1. Constructor : `Greeter(String aName)`
2. A method : `sayHello( )` (Java uses the term “method” for a function defined in a class.)
3. A field : `name`

“Hello, World!” in Java

GreeterTest.java

```
public class GreeterTest
{
    public static void main(String[] args){
       Greeter worldGreeter = new Greeter("World");
        String greeting = worldGreeter.sayHello();
        System.out.println(greeting);
    }
}
```

Datatypes in Java

- Primitive types:
  - `int, long, short, byte, char, boolean, double, float`
  - Characters are encoded in “Unicode”
  - Conversions that don’t incur information loss (such as `short` to `int` or `float` to `double`) are always legal.
  - All other conversions require a cast such as:

```
double x = 10.0 / 3.0;        // x = 3.3333333333333335
int n = (int)x;               // sets n = 3
float f = (float)x;           // x = 3.3333333
```

Java Strings

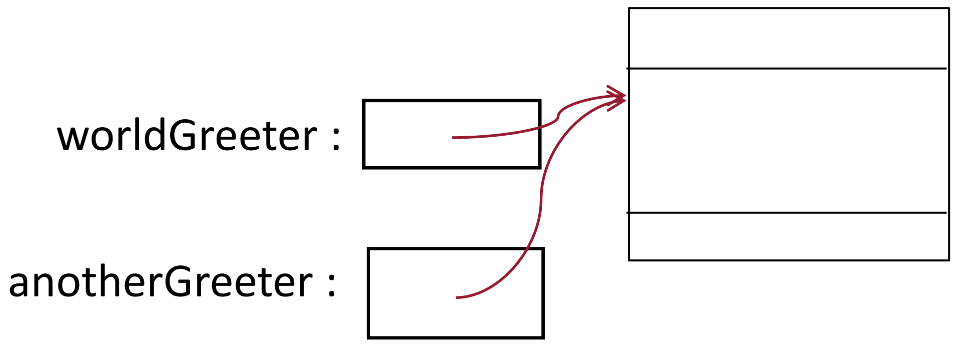
- Java strings are represented by objects of the String class.
- Java strings are sequences of Unicode characters.
- There are two ways to make a String object: you can use a string literal, or you can use a constructor.
  - `String greeting = "Hello";`
  - `String greeting = new String();`
    `greeting = “Hello”;`
- Java strings are immutable. Once created, a string cannot be changed.
- Since Java strings are objects, you interact with them through the interface defined by the String class.

Object References

- In Java, an object value is always a reference to an object, (i.e., a value that describes the location of the object)

```
Greeter worldGreeter =
        new Greeter("World");

Greeter anotherGreeter =
        worldGreeter;
```



- You can have multiple variables referencing to the same object.

Parameter Passing

- Java has no call by reference. Both primitive types and object references are passed by value.
- While a method can change the *state* of an object that is passed as a parameter, it can never update the *value* of any variable.

```
public void setLength(int n){
       n = name.length();
}
public void setGreeter(Greeter other) {
       other = new Greeter(“Earth”);
}
```

```
// these assignments
have no effect
outside the method
```

```
public void setName(Greeter other){
       other.name = this.name;
}
Greeter worldGreeter = new Greeter("World");
Greeter daveGreeter = new Greeter("Dave");
worldGreeter.setName(daveGreeter);
```

What is `worldGreeter.name` after calling `setName`?

```
public void setGreeter(Greeter other) {
        other = new Greeter(“Earth”);
}
Greeter worldGreeter = new Greeter("World");
Greeter daveGreeter = new Greeter("Dave");
worldGreeter.setGreeter(daveGreeter);
```

```
public void setName(Greeter other){
        other.name = this.name;
}
Greeter worldGreeter = new Greeter("World");
Greeter daveGreeter = new Greeter("Dave");
worldGreeter.setName(daveGreeter);
```

Java Arrays

- Java arrays are objects. They can hold sequences of arbitrary values.
  - `char[] letters = new char[26];`
  - `int[][] table = {{1, 2},{3, 4}};`
  - `Greeter[] greets = new Greet[5];`
  - Note: an empty array of length 0 (`new int[0]`) is different from `null` — a reference to no array at all.
- When an array is constructed, its elements are set to 0, false, or null.
- After an array has been constructed, you cannot change its length.
- If you access a nonexistent position (< 0 or >= length), then an `ArrayIndexOutOf-BoundsException` is thrown.
- You can store instances of subclasses in an array of the superclass.
  - The "`instanceof`" method can be used to test if an object is of a specified type.

Java ArraysLists

- ArrayList class lets you collect a sequence of objects of any type.
  - ArrayList cannot contain primitive datatypes.
  - Unlike arrays you can add new objects to the ArrayList using the "add" method.
    - `ArrayList <String> countries = new <String> ArrayList();`
    - `countries.add(“United States”)`
  - The `get` method returns the object at a given position.
    - Since the return type of get is "Object", you need to *cast* the returned type to the correct object type
  - The set method lets you overwrite an existing element with another:
    - `countries.set(0, "France");`
  - You can `insert` and `remove` objects in the middle of the array list.
    - `countries.insert(1, "Germany");`
    - `countries.remove(0);`
  - Other differences between Arrays and ArrayLists : Performance, multi-dimentional support.

Java ArraysLists

- Similar to `Array` class, you can store objects from different subclasses in an `ArrayList`
- Example:

```
public class Vehicle {
    protected String name;
}
public class Bus extends Vehicle {
    public Bus(String name) {  this.name=name; }
}
public class Car extends Vehicle {
    public Car(String name) { this.name=name; }
}
public class Main {
    public static void main(String[] args) {
        Car car = new Car("BMW");
        Bus bus = new Bus("MAN");
        ArrayList<Vehicle> list = new ArrayList<Vehicle>();
        list.add(car);
        list.add(bus);
   }
}
```

Static Fields and Methods

- A common use for the static keyword is to define constants:

```
public class Math1{
     ...
     public static final double E = 2.7182818284590452354;
     public static final double PI = 3.14159265358979323846;
 }
```

You can refer to these as `Math1.PI` and `Math1.E`

- To share a field among all objects of a class, you can declare the field as static:

```
public class Greeter{
     ...
     private static Random generator = new Random();
 }
```

✓ A static method is a method that does not operate on objects.
  - Static methods can access static fields but not instance fields—they don’t operate on an object.

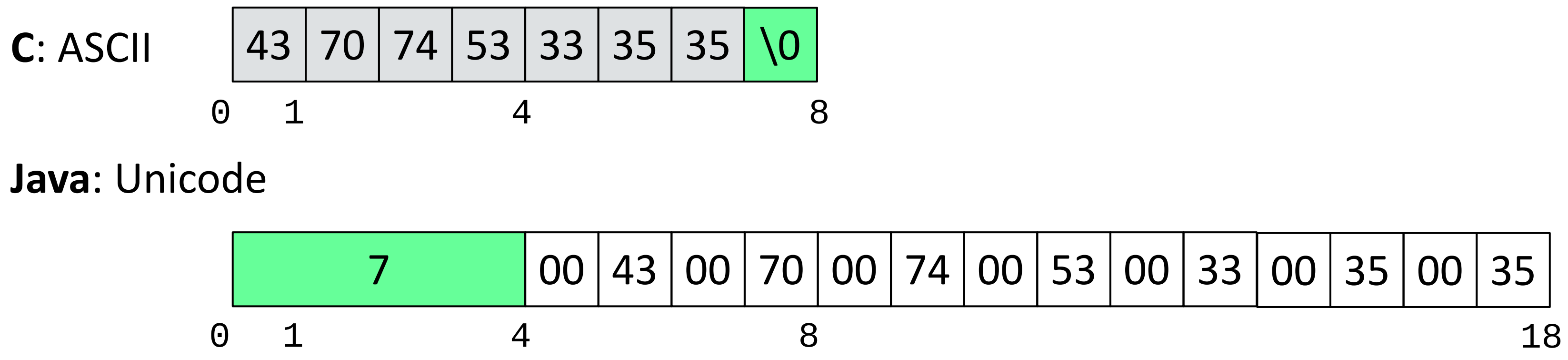
```
public static Greeter getRandomInstance(){
    if (generator.nextBoolean())
           return new Greeter("World");
    else
           return new Greeter("Mars");
}
```

You invoke this method as `Greeter.getRandomInstance()`.

Data Representations in Java

- Characters and strings
  - Unicode
    - represents most of the world's alphabets
  - String not bounded by a '\0' (null character)
    - Bounded by a hidden length field at the beginning of the string.

the string "CptS355"



Data Representations in Java

- Objects in Java vs structs in C
  - Java objects can only store primitive data types
  - Include complex data types (arrays, other objects, etc.) using references

C

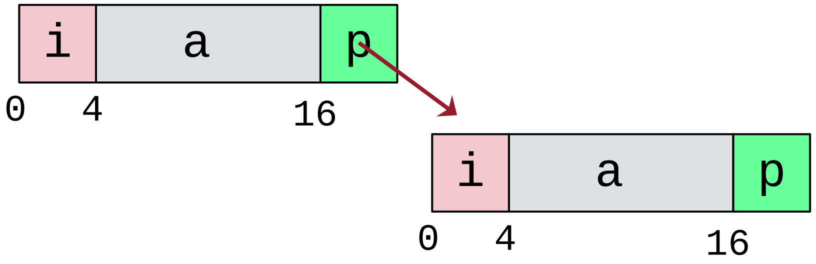
```
struct rec {
   int i;
   int  a[3];
   struct rec *p;
};
```

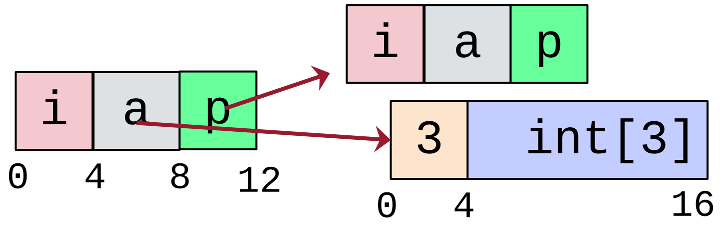
```
struct rec *r  = malloc(…);
struct rec *r2 = malloc(…);
r->i = val;
r->a[2] = val;
r->p = r2;
```

Java

```
class Rec {
   int i;
   int[] a = new int [3];
   Rec p;
…
};
```

```
r= new Rec();
r2= new Rec();
r.i = val;
r.a[2] = val;
r.p=r2;
```





Data Representations in Java

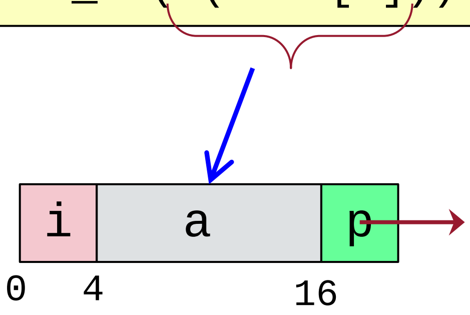
- Pointers/References
  - Pointers in C can point to any memory address
  - References is Java can only point to an object
    - And only to its first element – not to the middle of it.

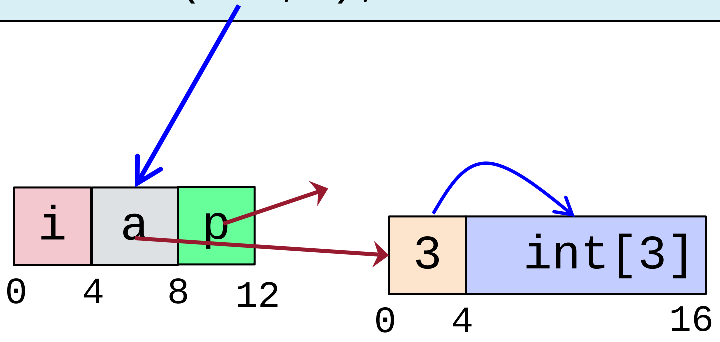
C

```
struct rec {
   int i;
   int  a[3];
   struct rec *p;
};
…
some_fn(&(r->a[1]))  //ptr
```

Java

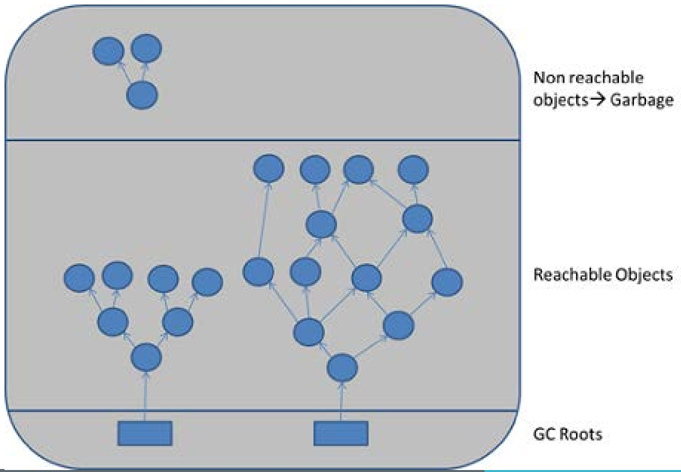
```
class Rec {
   int i;
   int[] a = new int [3];
   Rec p;
};
…
some_fn(r.a,1);  //ref and index
```





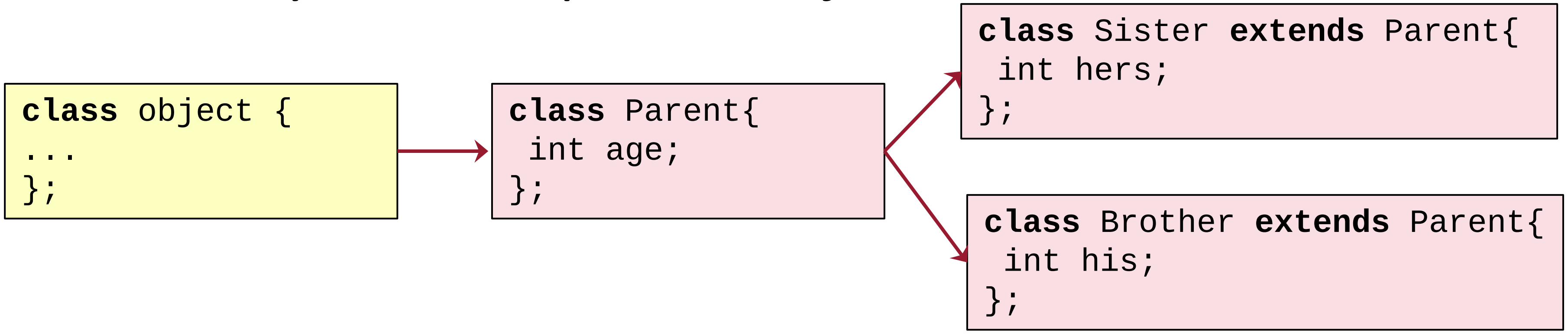
Garbage Collection

- The Java platform uses a garbage collector to automatically reclaim memory by recycling objects when they are no longer referenced.
  - The `malloc()` and `free()` functions used by C aren't necessary in Java programming, and no similar methods exist for the Java language.
- To determine which objects are no longer in use, the Java Virtual Machine occasionally runs a "mark-and-sweep" algorithm.



Casting in Java

- Can only cast compatible object references



```java
class object {
...
};
```

```java
class Parent{
 int age;
};
```

```java
class Sister extends Parent{
 int hers;
};
```

```java
class Brother extends Parent{
 int his;
};
```

```java
//Parent is a superclass of Brother and Sister, which are siblings

Parent a = new Parent();
Sister xx = new Sister();
Brother xy = new Brother();

Parent p1 = new Sister();
Parent p2 = p1;
Sister xx2 = new Brother();
Sister xx3 = new Parent();
Brother xy2 = (Brother) a;
Sister xx4 = (Sister)p2
Sister xx5 = (Sister)xy
```