ESS 201: Programming II
Java

Term 1, 2020-21

Classes

T K Srikanth
IIIT-B

Types in Java

- primitives
- reference
- Array
- (a special type) void

Primitive types

Fixed size of basic types

boolean true, false (not equivalent to 0 or 1)
char 16 bits (unicode)
byte 8 bits
short 16 bits
int 32 bits
long 64 bits
float 32 bits
double 64 bits

all numeric types are “signed”. No “unsigned”
Note: primitives **are NOT** objects

Java operators and control statements

Operators:

- Generally follow the syntax of C/C++
- No operator overloading

Control statements:

- If-else, while, do-while, for: similar to C/C++

for loop - additional syntax

```
for( T elem: <<array of T>>) {
 // elem is successively bound to each element of the array
}
```

E.g.

```
int[] scores = new int[400];
… initialize scores
int total = 0;
for (int score: scores) {
   total += score;
}
```

Useful when iterating through an array, but not modifying array or its elements

switch statement

variable being tested in switch can evaluate to constants of byte, short, char, and int primitive data types

Can also be used with enum

As well as String

Objects and references

- Everything is an “object” - an instance of a “class”
- A class encapsulates “behaviour” and “state”
  - Objects of a class exhibit the same set of behaviour, but can have different state values
  - Of course, exact action of a behaviour will depend on the state
- You manipulate objects through their references
- Objects (state) may change, but references do not
- Methods are passed references to objects (or primitives), so always “call by value”
- Objects are allocated on the heap, references are on the program stack or heap (depending on whether they are local variables or members of objects)
- Memory of objects are “garbage collected” when no longer referenced (How?)

String and Array

Both are **classes**

Specific methods, constructors, operations

String: immutable in size and content - can only be queried

Operators such as concatenation (+) defined.

Array:

Size defined at construct time (= new Type[size])

Can be queried for length (arr.length)

Class definition

```
class IceCreamBar {

  String getName()     {   }

  String getFlavour()  {   }

  float getTemp()      {   }

  float getPrice()     {   }

   ….

}
```

Methods or “behaviour”

Entire definition of a class - all its methods and data members - are in a single file, whose name is <classname>.java. So, this definition would be in a file: IceCreamBar.java

Class definition

```
class IceCreamBar {

  String getName()     {   }

  String getFlavour()  {   }

  float getTemp()      {   }

  float getPrice()     {   }

  String name, flavour;

  float temp, price;

}
```

Methods or “behaviour”

Data members or **“State”**

Class definition

```
class IceCreamBar {
  IceCreamBar(String iname, String iflavour, float itemp,
              float iprice) { }

  String getName()    {  }

  String getFlavour() {  }

  float getTemp()     {  }

  float getPrice()    {  }

  String name, flavour;

  float temp, price;

}
```

Constructor

Methods or “behaviour”

Data members or **“State”**

Constructors

Every class should have at least one constructor

Constructor that takes no arguments is called the *default constructor*

Compiler defines a default constructor if no constructors have been defined for the class. This default version sets all data members to their *default initial value*.

All instance variables are initialized to default values, if not explicitly initialized in a constructor

If any constructor is defined for the class, then the constructor does not define the default constructor

Note: no notion of "*destructors*". Facility for cleanup

Class definition

```
class IceCreamBar {

  IceCreamBar(String iname, String iflavour, float itemp,
float iprice) { … }

  String getName() { return name; }

  String getFlavour() { return flavour; }

  float getTemp() { return temp; }

  float getPrice() { return price; }

  …

}
```

Invoking class methods

```
IceCreamBar ic1 = new IceCreamBar("CH", "Chocolate", 8.0,90.0);

    …

System.out.println("Flavour " + ic1.getFlavour() +
                                " from " + ic1.getName() +
                                "costs " + ic1.getPrice() +
                                " and is stored at " +
                                ic1.getTemp() + " degrees.");
```

Should print:

```
Flavour Chocolate from CH costs 90.0 and is stored at 8.0
degrees.
```

Note: any invocation of a method of some class always is from within some method of the same or other class. No free standing executions/statements. Even **main** is a method of some class

Class definition

```
class IceCreamBar {
  IceCreamBar(String iname, String iflavour, float itemp,
float iprice) {
        name = iname; flavour = iflavour;
        temp = itemp;
        price = iprice;
  }
  String getName() { return name; }
  String getFlavour() { return flavour; }
  float getTemp() { return temp; }
  float getPrice() { return price; }
…
}
```