2 Language Basics – Expressions, Statements, Blocks

Statements

- Statement: Complete unit of execution (ends with **)
- Expression statements:
  - Assignment expressions: \( a = (17 + (3 * 5)) \% 3 \)
  - Use of ++ or --: \( a++; \)
  - Method invocations: \( someObject.methodOne() \)
  - Object creation expressions: \( new SomeClass() \)
- Declaration statements: \( int \ a; \)
- Blocks:
  - (next slide)
- Control flow statements
  - (later)

2 Language Basics – Control Flow Statements

- if and if else have a straightforward meaning:
  ```java
  void applyBrakes()
  {
      if (speed > 10) {
          speed = speed - 1;
      }
  }
  ```
- else:
  ```java
  void applyBrakes()
  {
      if (speed > 10) {
          speed = speed - 2; // break really hard
      } else if (speed > 0) {
          speed--; // soft brakes
      }
      System.err.println("The bicycle has already stopped!");
  }
  ```
- switch: Equivalent to sequence of chained if else statements
2. Language Basics – Control Flow Statements

- if and if else have a straightforward meaning:

```java
void applyBrakes()
{
    if (speed > 10) {
        speed = speed - 1;
    }
}
```

```java
void applyBrakes()
{
    if (speed > 10) {
        speed = speed - 2; // break really hard
    } else if (speed > 0) {
        speed--; // soft brakes
    }
    System.err.println("The bicycle has already stopped!");
}
```

- switch: Equivalent to sequence of chained if else statements

- for: usually means to do something for a fixed number of times:

```java
for (int i = 0; i < 7; i++) // loop will be executed 7 times
    System.out.println("i = "+i);
```

General form:

```java
for (initialization; termination; update) {
    statement;
}
```

- initialization expression: Executed once at the beginning of first loop
- termination expression: If true then execute statement(s), else exit loop
- update expression: Executed after each iteration of the loop
```java
package Demo;

public class Demo {
    public static void main(String[] args) {
        long fact = 1;
        while (n > 0) {
            result *= n;
            n = n - 1;
        }
        return result;
    }

double power(double x, int n) {
    double result = 1.0;
    for (int i = 0; i < n; i++)
        result = result * x;
    return result;
}

double square(double x) {
    return power(x, 2);
}

double product(double[] a, double[] b) {
    double result = 0.0;
    for (int i = 0; i < a.length; i++)
        result += a[i] * b[i];
    return result;
}
```
3 Classes, Objects, Inheritance

Recommended reading:

http://docs.oracle.com/javase/tutorial/java/javaOO/classes.html
http://docs.oracle.com/javase/tutorial/java/javaOO/objects.html
http://docs.oracle.com/javase/tutorial/java/javaOO/more.html
http://docs.oracle.com/javase/tutorial/java/lang/subclasses.html
http://docs.oracle.com/javase/tutorial/essential/exceptions/index.html

```java
class Bicycle {
    public int cadence = 0;
    public int speed = 0;
    public int gear = 1;

    public Bicycle(int startCadence, int startSpeed, int startGear) {
        gear = startGear;
        cadence = startCadence;
        speed = startSpeed;
    }

    public void changeCadence(int newValue) {
        cadence = newValue;
    }

    public void changeGear(int newValue) {
        gear = newValue;
    }

    public void applySpeed(int newValue) {
        speed = newValue;
    }
}
```

```java
public class MountainBike extends Bicycle {
    public int seatHeight;

    public MountainBike(int startHeight, int startCadence, int startSpeed, int startGear) {
        super(startCadence, startSpeed, startGear);
        seatHeight = startHeight;
    }

    public void setHeight(int newValue) {
        seatHeight = newValue;
    }
}
```

Source: [Tutorial]
3 Classes, Objects, Inheritance

Class Bicycle

```java
public class Bicycle {
    public void changeGear(int newGear) {
        gear = newGear;
    }
    public void speedUp(int newSpeed) {
        speed = newSpeed;
    }
    public void applyBrakes(int newBrakes) {
        brakes = newBrakes;
    }
}
```

- **Field declaration (general form):**
  
  ```java
  modifier type name;
  ```

  - **(Access) modifier (for fields):**
    certain combinations of {public, protected, private, static, final}

  - **type:** Any primitive or reference type

- **Class definition (general form):**
  
  ```java
  modifier class MyClass extends MySuperClass
  implements YourInterface1, ..., YourInterfaceN
  {
      // fields, constructors, methods
  }
  ```

  - **(Access) modifier (for classes):**
    certain combinations of {public, protected, private, static, final, abstract}

Method declaration (general form):

```java
modifier typeOfReturnValue name ( parameter* ) throwsClause { 
    statement*
}
```
3 Classes, Objects, Inheritance

- Field declaration (general form):
  ```java
  modifier type name;
  ```
- (Access) modifier (for fields):
certain combinations of {public, protected, private, static, final}
- type: Any primitive or reference type

- Method declaration (general form):
  ```java
  modifier returnType name (parameter*) throwsClause {
  statement*
  }
  ```
- (Access) modifier (for methods):
certain combinations of {public, protected, private, static, final, abstract}
- returnType: Any primitive or reference type
- parameter*: (later)
- throwsClause*: (later)
- statement*: statement(s) to execute

- Public int setHeight(int newValue) {
  seatHeight = newValue;
}

Source: [Tutorial]