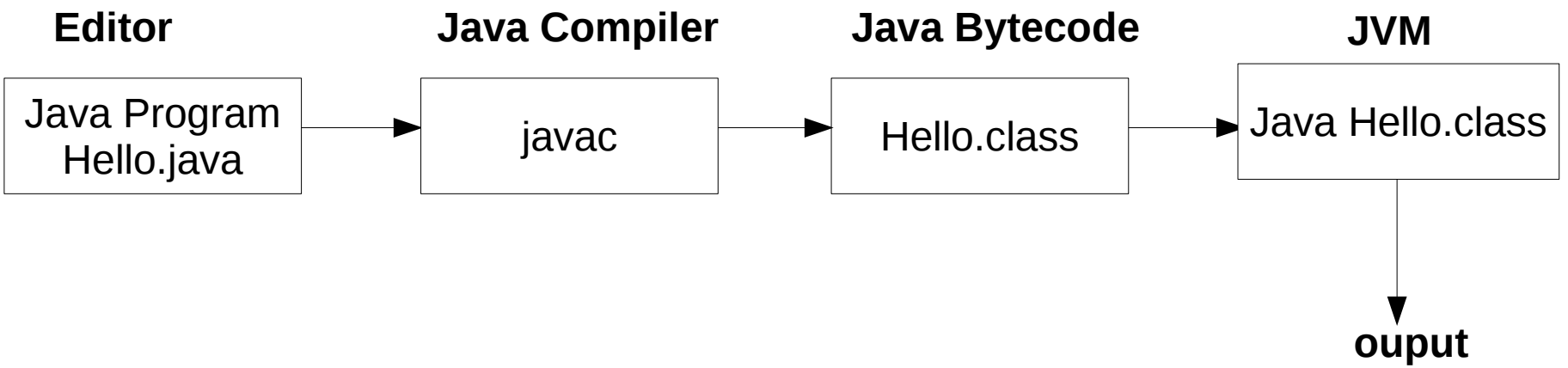
A Brief Introduction to Programming in Java

Your First Program

- Programming in Java: To program in Java you need three steps:
  1) *Creating a program*. Create a program by typing it into a file named and stored in a file with a .java extension, say, **Hello.java**.
  2) *Compile* it by typing **javac Hello.java** in a terminal window. The compiler takes the file with a .java extension as input (i.e., MyCode.java) and produces a **Bytecode** file with the same name but with a .class extension (i.e., Mycode.class)
  3) *Run* (or *execute*) it by typing **java Hello** in the terminal window. The Bytecode can be executed on any computer that has *Java Virtual Machine* (**JVM**). To use the **JVM** to execute your program, type the java command followed by the program name in a terminal window

Developing a Java program



OOP concepts

- **object-oriented programming**: In an OOP, a software system is represented as a collection of objects that interact with each other to solve the overall problem.
- **class/object**: A class is a type of a template definition of the methods and fields (variables), and is the . An object is is an instance of a class and combines data with behavior that acts on that data.
- **Inheritance**: The ability for a class ("subclass") to inherit commonly used state and behavior or **override** functionality of another class ("superclass")
- **polymorphism**: The ability to replace an object with its sub-objects to achieve different behavior from the same piece of code.
- **interface**: A specification of method signatures without supplying implementations, as a mechanism for enabling polymorphism.

OOP concepts

- **Fields**: A variable inside an object that is used to store its state.

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

- **object methods:** Operates on an object's internal states and gives behavior to each object.

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

- **Constructors:** create and initialize the state of a new object.

```
public Bicycle(){

}
```

OOP concepts

- **Inheritance**: Forming new classes based on existing ones.

Superclass: Parent class being extended

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

    public Bicycle( int startSpeed, int startGear) {
        gear = startGear;
        speed = startSpeed;
    }
...
}
```

Subclass: child class that inherits behavior from parent class. A subclass can call its parents method/constructors using super key word

```
public class Mountainbike extends Bicycle {
        public int seatHeight;
        public Mountainbike(int startHeight,int startSpeed,
int startGear) {
    super(startSpeed, startGear);
            seatHeight = startHeight;
        }

...

}
```

Oregon State University

OOP concepts

- **Interface**: Forming new classes based on existing ones without code sharing. In other words, an interface is a group of related methods with empty bodies.

```
public interface Shape {
    public abstract double area();
    public abstract double perimeter();
}
```

```
public class Rectangle implements Shape {
    private final double width, length; //sides

    public Rectangle() {
        this(1,1);
    }
    public Rectangle(double width, double length) {
        this.width = width;
        this.length = length;
    }
    public double area() {
        return width * length;
    }
...
```

```
public class Circle implements Shape {
    private final double radius;
    final double pi = Math.PI;

    public Circle() {
        this(1);
    }
    public Circle(double radius) {
        this.radius = radius;
    }
    public double area() {
        return pi * Math.pow(radius, 2);
    }
...
```

OOP concepts

- **Abstract class**: Abstract classes allow you to define some behaviors (methods), and they force your subclasses to provide others.
- **Polymorphism**: Abstract and interface classes allow to implement polymorphism (the same code can work with different types of objects)

```
public abstract class GraphicObject {
    int x, y;
    ...
    void moveTo(int newX, int newY) {
        ...
    }
    abstract void draw();
    abstract void resize();
}
```

```
public class Circle extends GraphicObject{
void draw() {
        ...
    }
   void resize() {
        ...
    }
}
```

```
class Rectangle extends GraphicObject {
    void draw() {
        ...
    }
    void resize() {
        ...
    }
}
```

Oregon State University

References:
Intro to Java Programming, Comprehensive Version by Y. Daniel
https://courses.cs.washington.edu/courses/cse331/11sp/lectures/slides/
https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html