COMP 110 AND 401
CLASS (STATIC) STATE

Instructor: Prasun Dewan
Prerequisites

- State and Properties
- Interfaces
COUNTING INSTANCES OF A CartesianPoint

The image shows a user interface for creating a Cartesian point. The parameters for creating the point are set to:
- Parameter 1: int, value 25
- Parameter 2: int, value 50

The resulting instance is displayed in a separate window, showing the point's properties:
- Num Instances: 1
- Angle: 1.107148717
- Radius: 55.9016994
- X: 25
- Y: 50
INCREMENTSED NUMBER OF INSTANCES

Parameters of Create ACartesian Point

Parameter 1: int 125
Parameter 2: int 150
Create ACartesian Point

ACartesianPoint, [ACartesianPoint]

Num Instances: 2
Angle: 0.876058050
Radius: 195.256241
X: 125
Y: 150
CREATING MIDPOINT
SPECIFYING TWO END POINTS
TYPE OF METHODS

- Difference between method returning number of instances and other methods seen so far?
- Difference between mid and other methods seen so far?
numCarsProduced();

new ACorvette(silver)

new ACorvette(red)

blend(silverCar, redCar);

101234

getCodeage() 64000

numCarsProduced();
new ACartesianPoint(25, 50)
new ACartesianPoint(125, 150)

getRadius()

mid(25, 50, 125, 150);

numInstances() 3

ACartesianPoint

mid(point1x, point1y, point2x, point2y);
CLASS METHODS

- Methods can be invoked on class itself
- Called class or static methods
- Declared in class on which they are invoked
- Keyword static in header
- Accesses no instance variable
PROGRAMMER-DEFINED MID

### Instance Method

```java
public getRadius (){
    return Math.sqrt(x*x + y*y);
}
```

### Access instance variables

### Class Method

```java
public static Point mid (int x1, int y1, int x2, int y2 ) {
    return new ACartesianPoint(x1 + (x2 - x1)/2, y1 + (y2 - y1)/2);
}
```

### Access no instance variable
EXTERNAL CALL OF CLASS METHOD

ACartesianPoint.mid(x1, y1, x2, y2)

Class as target

Math.sqrt(x*x + y*y);
numInstances Algorithm

Declare variable, numInstances initialized to zero

Increment numInstances each time a new instance is created

getNumInstances() returns numInstances
RETURNING numInstances

public static int getNumInstances() {
    return numInstances;
}

numInstances() returns numInstances

Class property
Incrementing numInstances

Increment numInstances each time a new instance is created
Incrementing numInstances (Solution)

```java
public ACartesianPoint(int theX, int theY) {
    x = theX;
    y = theY;
    numInstances++;
}
```

Increment numInstances each time a new instance is created
Declaring numInstances

Declare variable, numInstances initialized to zero

static int numInstances = 0; // class variable
public ACartesianPoint(int theX, int theY) {
    x = theX;
    y = theY;
    numInstances++;
}
**INSTANCE VS. CLASS MEMBERS**

- **Class Members**
  - Class method
  - Class variables

- **Instance Members**
  - Instance methods
  - Instance variables
SCOPE OF CLASS AND INSTANCE MEMBERS

Class Members
- visible to other class members
- visible to all instance members
  - class & instance methods can access class variables
  - class and instance methods can call class methods

Instance Members
- visible to other instance members
- not visible to class members
- which of (zero to many) copies of an instance variable should a class member refer to?
static int numInstances = 0;

public static int getNumInstances() {
    System.out.println(x);
    return numInstances;
}

public ACartesianPoint(int theX, int theY) {
    x = theX; y = theY;
    numInstances = numInstances + 1;
    System.out.println(getNumInstances());
}

public double getRadius() {
    return Math.sqrt(x*x + y*y);
}
**INSTANCE vs. CLASS NAMED CONSTANT**

```java
public class AnotherBMISpreadsheet implements BMISpreadsheet{
    double height, weight, bmi;
    ...
    public static final double LBS_IN_KG = 2.2;
    public static final double CMS_IN_INCH = 2.54;
    double calculateBMI() {
        return weight / LBS_IN_KG / (height * CMS_IN_INCH / 100) / (height * CMS_IN_INCH / 100);
    }
}
```

```java
new AnotherBMISpreadsheet();
AnotherBMISpreadsheet.LBS_IN_KGS;
```
**INSTANCE vs. CLASS NAMED CONSTANT**

```java
public class AnotherBMISpreadsheet implements BMISpreadsheet{
    double height, weight, bmi;
    ...
    public final
        double LBS_IN_KG = 2.2;
    public static final
        double CMS_IN_INCH = 2.54;
    double calculateBMI() {
        return (weight/LBS_IN_KG) /
        (height*CMS_IN_INCH/100*height*CMS_IN_INCH/100);
    }
}
```

```java
new AnotherBMISpreadsheet()
new AnotherBMISpreadsheet()
(new AnotherBMISpreadsheet()).LBS_IN_KGS;
```
CLASS VS. INSTANCE CONSTANT

- Should be class constant
  - one copy
  - easy to refer (require no instance creation)
- Unless some good reason for hiding named constants from static methods
Class Method

```java
public static Point mid(int x1, int y1, int x2, int y2) {
    return new ACartesianPoint(x1 + (x2 - x1)/2, y1 + (y2 - y1)/2);
}
```

Instance Method

```java
public Point mid(int x1, int y1, int x2, int y2) {
    return new ACartesianPoint(x1 + (x2 - x1)/2, y1 + (y2 - y1)/2);
}
```

- ACartesianPoint.mid(25, 50, 125, 150)
- Math.round(5.7)

Accesses no instance variable

(new ACartesianPoint(25, 50)).mid(25, 50, 125, 150)
(new Math()).round(5.7)
Class vs. Instance Method

- Instance method has all the privileges of a class method
- Any class method can be made an instance method
- Bad style to have instance method that does not access any instance variable
  - They belong to the class
  - Violate least privilege principle
  - Require needless instantiation
CLASS VS. INSTANCE CONSTANT/METHOD

- Named constants should be static
  - Unless some good reason for hiding named constants from static methods
- Methods not accessing instance variables should be static
  - Unless need to be listed in interface
CLASS MEMBER RESTRICTIONS

- Cannot define interfaces for classes
  - No conceptual reason why not
  - Non object oriented languages supported them
  - Class members go against the idea of OO programming (not instantiated) so treated as second class.

- Cannot use super in class methods
  - Can name super class directly
  - Non dynamic dispatch (discussed later)
Revisit, through non-graphical objects, concepts illustrated in previous sections
public Loan add(Loan loan2) {
    return new ALoan(getPrincipal() + loan2.getPrincipal());
}

public Loan getTotalLoan() {
    return houseLoan.add(carLoan);
}
**Class (Static) Add**

**Instance Method**

```java
public Loan getTotalLoan(){
    return ALoan.add(houseLoan, carLoan);
}
```

**Class Method**

```java
public static Loan add(Loan loan1, Loan loan2) {
    return new ALoan(loan1.getPrincipal() + loan2.getPrincipal());
}
```

**Accesses instance variables**

**Access no instance variable**
**Non-Polymorphic Methods**

```java
public static Loan add(ALoan loan1, ALoan loan2) {
    return new ALoan(loan1.getPrincipal() + loan2.getPrincipal());
}

public static Loan add(AnotherLoan loan1, AnotherLoan loan2) {
    return new ALoan(loan1.getPrincipal() + loan2.getPrincipal());
}

public static Loan add(ALoan loan1, AnotherLoan loan2) {
    return new ALoan(loan1.getPrincipal() + loan2.getPrincipal());
}
```

*Code duplication!*
**OVERLOADING VS. POLYMORPHISM**

```java
public static Loan add(Loan loan1, Loan loan2) {
    return new ALoan(loan1.getPrincipal() + loan2.getPrincipal());
}

public static Loan add(ALoan loan1, ALoan loan2) {
    return new ALoan(loan1.getPrincipal() + loan2.getPrincipal());
}

public static Loan add(ALoan loan1, AnotherLoan loan2) {
    return new ALoan(loan1.getPrincipal() + loan2.getPrincipal());
}
```

```java
add (new ALoan(10000), new ALoan(5000));
add (new ALoan(10000), new AnotherLoan(5000));
```

---

**Polymorphism**

**Overloading**
PRIMITIVE VS. OBJECT TYPES

types

Primitive types

double
int

Object types

Classes
ABMICalculator
ABMISpreadsheet
ALoan
AnotherLoan

Interfaces
BMISpreadsheet
Loan

type = set of operations
REAL-WORLD ANALOGY (O-O PROGRAMMING)

numCarsProduced();

101234

new ACorvette(silver)

new ACorvette(red)

blend(silverCar, redCar);

getMileage() 64000

O-O programming with class methods and variables

Blueprints as classes would not have modelled class state and operations
new ACartesianPoint(25, 50)

new ACartesianPoint(125, 150)

mid(point1x, point1y, point2x, point2y);

mid(25, 50, 125, 150);

O-O programming with class methods and variables

numInstances()
GENERAL CLASS STRUCTURE

- Class Var 1
- Class Var 2
- Instance Var 2
- Instance Var 2
- Class Method 1
- Class Method 2
- Instance Method 1
- Instance Method 2
Example of Java Class?
CLASS WITH NO INSTANCE MEMBERS?

Class

Class Var 1
Class Var 2

Class Method 1
Class Method 2

Creating instances meaningless
Example of Java class?
public static final double E

public static final double PI

public static double log(double a)

public static double abs(double a)
public static final InputStream in

public static final PrintStream out

public static void exit(int status)

public static long currentTimeMillis()

Real-world analogy?
Real Life Analogy

Warehouse, dealership
CLASS WITH ONLY CLASS METHODS?

Class

Class Method 1

Class Method 2

Example?
public static final double E

public static final double PI

public static double log(double a)

public static double abs(double a)
MODIFIED MATH

Math

public static double abs(double a)

Real-world analogy?
REAL LIFE ANALOGY: PURE SERVICE

Assume nothing stored
CLASS WITH ONLY CLASS VARIABLES?

Class Var 1

Class Var 2

Class

Example?
public static final InputStream in

public static final PrintStream out

public static void exit(int status)

public static long currentTimeMillis()
**Modified System**

```java
public static final InputStream in
public static final PrintStream out
```

Real-world analogy?
UNGUARDED AIR HOSE
**Class vs. Instance Object**

Should some state and associated methods be accessed as a class or instance object?

- **As class object**
  - Multiple independent instances of that state will not occur.
  - Do not need dynamic dispatch or interfaces to that state.

  **E.g:** System.in

- **Singleton class**
  - A class with only one instance, useful when dynamic dispatch or interface needed.
Franchise: Multiple franchisees exist, so corresponds to an instance object
**Class or Instance?**

One of, corresponds to a class object

Mom and pop store can later become franchise as it becomes popular and evolves

Class objects often converted to instance objects as program becomes popular and evolves

When in doubt make it an instance object