**Review Session**

**Topics**
- Invariants
- Subclasses & Constructors
- Abstract

---

**Why Class Invariants?**
- A class invariant is a condition that defines all valid states for an object.
- It is a logical condition to ensure the correct working of a class.
- Class invariants must hold when an object is created, and they must be preserved under all operations of the class.
- In particular all class invariants are both preconditions and post-conditions for all operations or member functions of the class.

---

**Example**

```java
public class Time {
    private int hr;
    private int min;

    /** Constructor: an instance with time t, in minutes, in range 0..60 */
    public Time(int t) {
        hr= t / 60;
        min= t % 60;
    }
}
```

---

**Constructors??**
- A **Java constructor** has the same name as the name of the class to which it belongs. Constructor's syntax does not include a return type, since constructors never return a value.
- Constructors may include parameters of various types. When the constructor is invoked using the new operator, the types must match those that are specified in the constructor definition.
- Java provides a default constructor which takes no arguments and performs no special actions or initializations, when no explicit constructors are provided.
Constructor Overloading

- public Organism(int lev, int m, String nn)
  { … }
- public Organism(int lev) { … }

So, the user can write new Organism(4) instead of new Organism(4, 0, null).

Which is the correct form?

1. public Organism(int lev) {
   Organism(lev, 0, null);
}

2. public Organism(int lev) {
   this(lev, 0, null);
}

Example - fall 2006

Question 5 (21 points). Consider the classes provided below and answer the following questions.

(a) In class Positive, write the body of the constructor.
(b) In class Rational, write the bodies of the constructor and procedure setPositive. In doing these, keep in mind that the rational number must always be maintained with the denominator > 0 and in lowest possible terms. E.g. the rational number 15/45 is maintained as 1/3 and 25/15 as 5/3.
(c) Explain why class Rational overrides procedure setPositive

Solution

/* An instance wraps a positive integer */
public class Positive{
  private int k; // the positive integer

  /** Constructor: an instance with value k. */
  public Positive (int k) {
    this.k = k;
  }

  /** = this instance's value */
  public int getPositive() {
    return k;
  }

  /** Set the value of this instance to n. */
  public void setPositive(int n){
    k = n;
  }
}
(b) In class Rational, write the bodies of the constructor and procedure setPositive. In doing these, keep in mind that the rational number must always be maintained with the denominator \(> 0\) and in lowest possible terms; e.g., the rational number \(15/45\) is maintained as \(1/3\) and \(25/15\) as \(5/3\).

```
public class Rational extends Positive {
    private int num;
    public Rational(int num, int denom) {
    }
    public void setPositive(int n){
    }
    public void reduce(){
    }
}
```

Subclasses

- The ability to extend existing subclasses to reuse/refine existing behavior is a terrific aspect of object-oriented programming.

Abstract Class?

- Why make a class abstract?

Definition

- An **abstract class** is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.
- An **abstract method** is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:
  ```java
  abstract void moveTo(double deltaX, double deltaY);
  ```

Note

- If a class includes abstract methods, the class itself must be declared abstract, as in:
  ```java
  public abstract class GraphicObject {
    // declare fields
    // declare non-abstract methods
    abstract void draw();
  }
  ```
- When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, the subclass must also be declared abstract.
Questions.