COMP 110-001
Designing Methods and Overloading

Yi Hong

June 04, 2015

Today

- Review of Constructors and Static Methods
- Designing methods
- Overloading methods

Example: Pet Class

```
public class Pet
{
    private String name;
    private int age;
    private double weight;

    public Pet()
    {
        name =  “No name yet” ;
        age = 0;
        weight = 0;
    }

    public Pet(String initName, int initAge, double initWeight)
    {
        name = initName;
        age = initAge;
        weight = initWeight;
    }
}
```

Constructors Self-test Questions

- If a class is named Student, what name can you use for a constructor of this class?
  - Every constructor for this class must be named Student
- What return type do you specify for a constructor?
  - No return type, not even void
- What is a default constructor?
  - Constructor without parameters

static, Some Examples

- static constants and variables
  - private static final int FACE_DIAMETER = 200;
  - public static final int FEET_PER_YARD = 3;
  - private static int numberOfInvocations;
- static methods
  - public static void main(String[] args)
  - public static int pow(int x, int y)

static Version of Pow Method

```
public class MathUtilities
{
    // Returns x raised to the yth power, where y >= 0
    public static int pow(int x, int y)
    {
        int result = 1;
        for (int i = 0; i < y; i++)
        {
            result *= x;
        }
        return result;
    }
}
```

static keyword

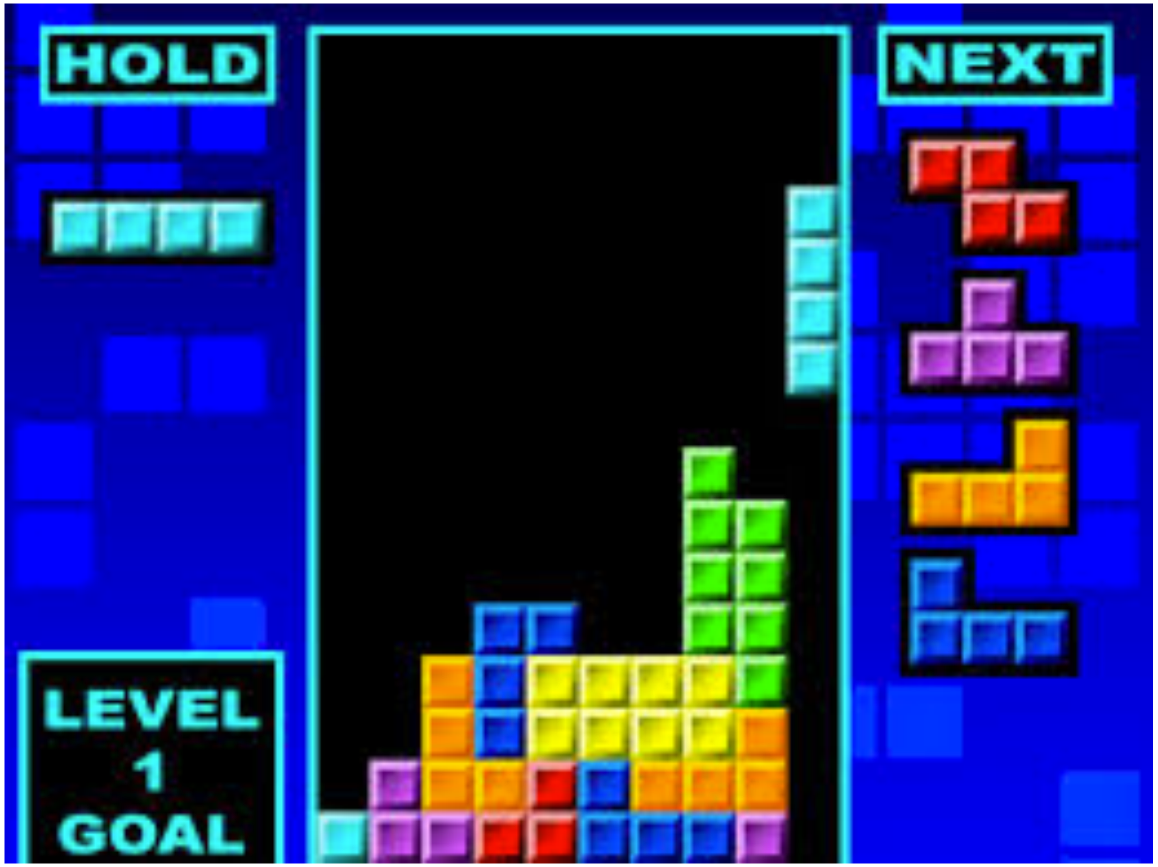
Static Self-test Questions

- Can you call a non-static method from a static method?
  - No, unless you first create an object of that class and use that object to invoke the non-static method

- Can you call a static method from a non-static method?
  - Yes!

- Can you access an instance variable inside a static method?
  - No!

Example of Designing Method: Tetris



Divide Task Into Small Groups

- Decide what high-level tasks are required for Tetris gameplay to work

- Assume the graphical display code is taken care of for you

Tetris High-Level Gameplay Tasks

- Choose a random tetromino to give the user
- User-controlled tetromino manipulation
- Game-controlled tetromino manipulation (automatically falling)
- Remove full horizontal lines of blocks
- Increase user score, level, and speed of falling blocks
- Check if game is over

User-Controlled Tetromino Manipulation

- High-level task: manipulate tetromino based on user input
- How can a tetromino be manipulated?
  - Move
  - Rotate

Moving a Tetromino

- How?

- Subtasks
  - Move left
  - Move right
  - Move down

Rotating a Tetromino

- Subtasks
    - Rotate clockwise
    - Rotate counterclockwise

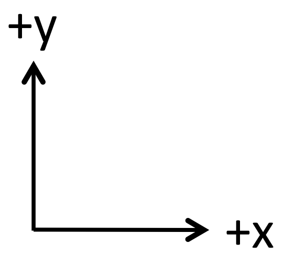
Design a Tetromino Class

```
public class Tetromino
{
    private int x;
    private int y;
    // some other stuff describing this Tetromino’s shape

    public void moveLeft()
    {
        x--;
    }

    public void moveRight()
    {
        x++;
    }

    public void moveDown()
    {
        y--;
    }
}
```



Top-Down Design

- Divide and conquer

- Start with a big problem
- Decompose problem into smaller subtasks
- Decompose big subtasks into even smaller subtasks
- Solve subtasks to solve big problem

Using the Tetromino Class in a Game Loop

```
public class TetrisGame
{
    private Tetromino userTetr;

    // gameUpdate() is called once per game loop
    public void gameUpdate()
    {
        // ...do some stuff here
        // check user input, assume userTetr has been properly
        // instantiated
        if (userInput == LEFT)
            userTetr.moveLeft();
        else if (userInput == RIGHT)
            userTetr.moveRight();
        else if (userInput == DOWN)
            userTetr.moveDown();

        applyAutoFalling(userTetr);

        // do some other stuff here
    }
}
```

Game-Controlled Tetromino Manipulation

- How can we implement automatically falling tetrominoes?

- What are we trying to do at a high level?
  - After an amount of time, make a tetromino move down one space
  - Need a timer

applyAutoFalling method

```
public void applyAutoFalling(Tetromino tetr)
{
    double timeSinceLastAutoFall =
        // some code to figure out the time since the last fall

    if (timeSinceLastAutoFall > 0.5)
    {
        tetr.moveDown();
    }
}
```

What if We See This Behavior?

- Imagine that we have run the game
  - A new tetromino appears
  - The user does not provide any input
  - The tetromino does not automatically fall, it simply stays where it is

- What could the problem be?

Let’s Check applyAutoFalling

```
public void applyAutoFalling(Tetromino tetr)
{
    double timeSinceLastAutoFall =
        // some code to figure out the time since the last fall

    if (timeSinceLastAutoFall > 0.5)
    {
        tetr.moveDown();
    }
}
```

- What if we had this code?

```
double timeSinceLastAutoFall = 0.0;
```

The problem could be elsewhere

- What if we had this code inside the class Tetromino?

```
public void moveDown()
{
    y = y;
}
```

- The moveDown() method does not do what it is supposed to do

Testing

- If a subtask (method) does not work, your solution is incorrect
- Test EVERY method you write

Bottom-Up Testing

- How do we determine if the error is in applyAutoFalling or moveDown?

- Test each method individually
  - If method A calls method B, fully test method B before testing method A
  - In this case, fully test moveDown before testing applyAutoFalling

Driver Programs

- Simple program for only you to test with
  - Run by you, not your user
- Call methods with different inputs
  - Test cases, edge conditions
    - Positive, negative, zero
    - true, false
    - Strings, characters
- Demonstrate MathUtils.java in Eclipse

Overloading

- Using the same method name for two or more methods *within the same class*

- We have seen this for constructors already

- Parameter lists must be different
  - public double average(double n1, double n2)
  - public double average(double n1, double n2, double n3)

- Java knows what to use based on the number and types of the arguments

Method *signature*

- A method’s name and the number and types of its parameters
- signature does NOT include return type
- Cannot have two methods with the same signature in the same class

Gotcha

- Overloading and automatic type conversion
- Imagine we have this constructor defined as:

```
public Pet(double initialWeight)
```

- We create a Pet like this:

```
Pet myPet = new Pet(35);
```

- What happens?

Gotcha

- Imagine we have these two constructors defined as:

```
public Pet(int initialAge)
public Pet(double initialWeight)
```

- We create a Pet like this:

```
Pet myPet = new Pet(35);
```

- What happens?

  We create a pet with age 35, instead of weight 35.0

Overloading and Polymorphism

- Self-test question 20 (p. 610): Is overloading a method name an example of polymorphism?
  - Answer on p. 654

20. This question may not have a definitive answer. In the original definition of *polymorphism*, overloading was considered an example of polymorphism, and some books still use that old definition. In current usage, and in this book, overloading a method name is *not* an example of polymorphism.

Next Class

- Package & Review of Classes