CPE207 Object Oriented Programming

Week 6

Deeper in classes: this() & toString()

Dr. Nehad Ramaha,
Computer Engineering Department
Karabük Universities

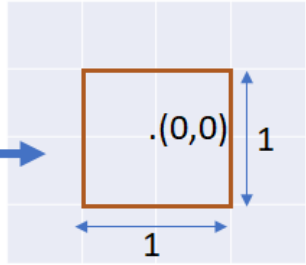
These Slides mainly adopted from Assist. Prof. Dr. Ozacar Kasim lecture notes

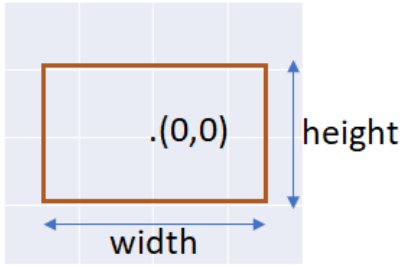
Table of Contents

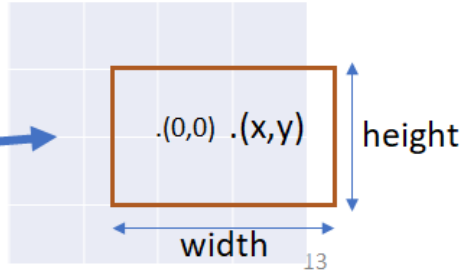
```java
public class Rectangle {
    private int x, y;
    private int width, height;

    public Rectangle() { //to create a unit rectangle(a square) at origin
        this.x = 0;        //user-defined  rectangle at specific coordinates.
        this.y = 0;
        this.width = 1;
        this.height = 1;
    }
    public Rectangle(int width, int height) { // to create user defined
        this.x = 0;  //user-defined  rectangle at specific coordinates.
        this.y = 0;
        this.width = width;
        this.height = height;


    }
    public Rectangle(int x, int y, int width, int height) { //to create a
        this.x = x;        //user-defined  rectangle at specific coordinates.
        this.y = y;
        this.width = width;
        this.height = height;
    }
}
```







13

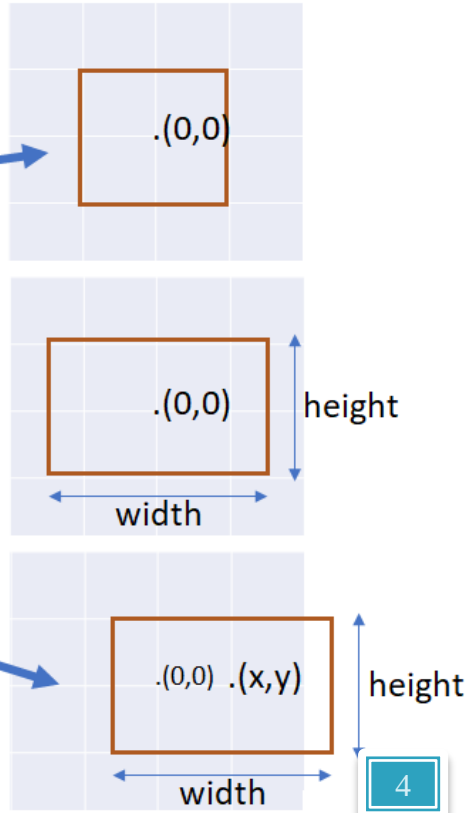
THIS WAY OF WRITING IS NOT GOOD

Calling a Constructor From a Constructor

this() constructors are used to call (invoke) an alternate constructor of the same class.

```
public class Rectangle {
    private int x, y;
    private int width, height;

    public Rectangle() { //to create a unit rectangle(a square) at origin
        this(0, 0, 1, 1);
    }
    public Rectangle(int width, int height) { // to create user-defined
        this(0, 0, width, height);   // rectangle at origin
    }
    public Rectangle(int x, int y, int width, int height) { //to create a
        this.x = x;        //user-defined  rectangle at specific coordinates.
        this.y = y;
        this.width = width;
        this.height = height;
    }
}
```



toString() method

- If you want to represent any object as a string, **toString() method** comes into existence.
- toString() returns a string representation of the object.
- In general, the toString method returns a string that "textually represents" this object.
- If you print any object, java compiler internally invokes the toString() method on the object.
- **We will mention it again in polymorphism.**

```
class Student{
 int rollno;
 String name;
 Student(int rollno, String name){
            this.rollno=rollno;
            this.name=name;
 }

public String toString(){
  return rollno+" "+name+" "+city;
 }

 public static void main(String args[]){
   Student s1=new Student(101,“Jack",“Sparrow");
   Student s2=new Student(102,“Johnny",“Cash");

   System.out.println(s1);
//compiler writes here s1.toString()
   System.out.println(s2);
//compiler writes here s2.toString()
 }
}
OUTPUT:
```

Garbage Collection

- Every object uses system resources, such as memory.
  - Wee need to give resources back to the system when they're no longer needed; **otherwise, "resource leaks" might occur.** (OutOfMemoryErrors)

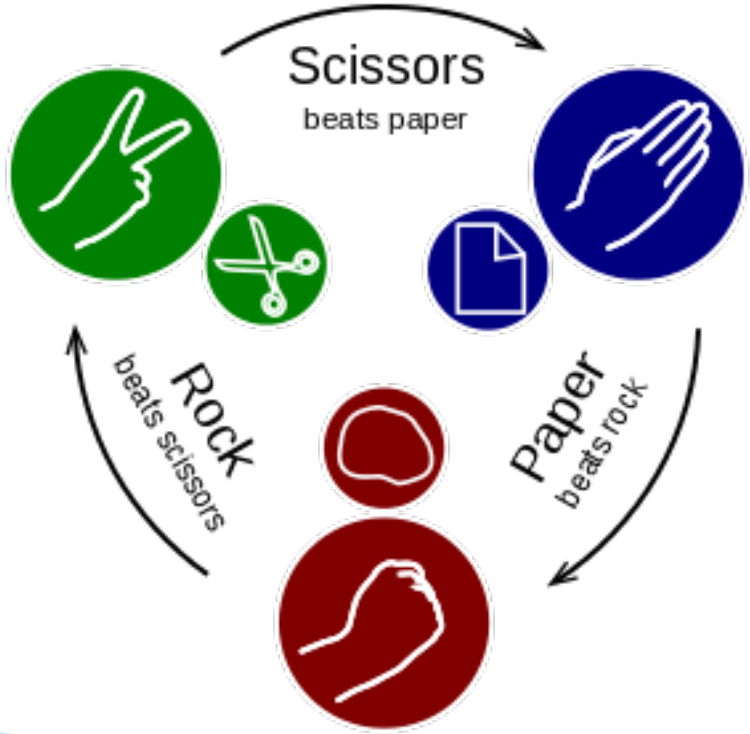
- In C/C++, programmer is responsible for both creation and destruction of objects.
  - [*int *ptr; ptr = (int *)malloc(sizeof(int)); *ptr = 25; free(ptr);*]

- But, The JVM performs automatic garbage collection to destroys the objects no longer in use.
  - When there are *no more references* to an object, the object is *eligible* to be collected. Collection typically occurs when the JVM executes its garbage collector.
  - Main objective of Garbage Collector is to free heap memory by destroying **unreachable objects.**

Garbage Collection (conts.)

- Every class in Java has the methods of class Object (package java.lang), one of which is method **finalize**.
- **finalize** allows the garbage collector to perform termination housekeeping on an object just before reclaiming the object's memory.
- You should *never* use method **finalize**, because it can cause many problems and there's uncertainty as to whether it will ever get called before a program terminates.

Yet Another Exercise: Rock Scissor Paper Game



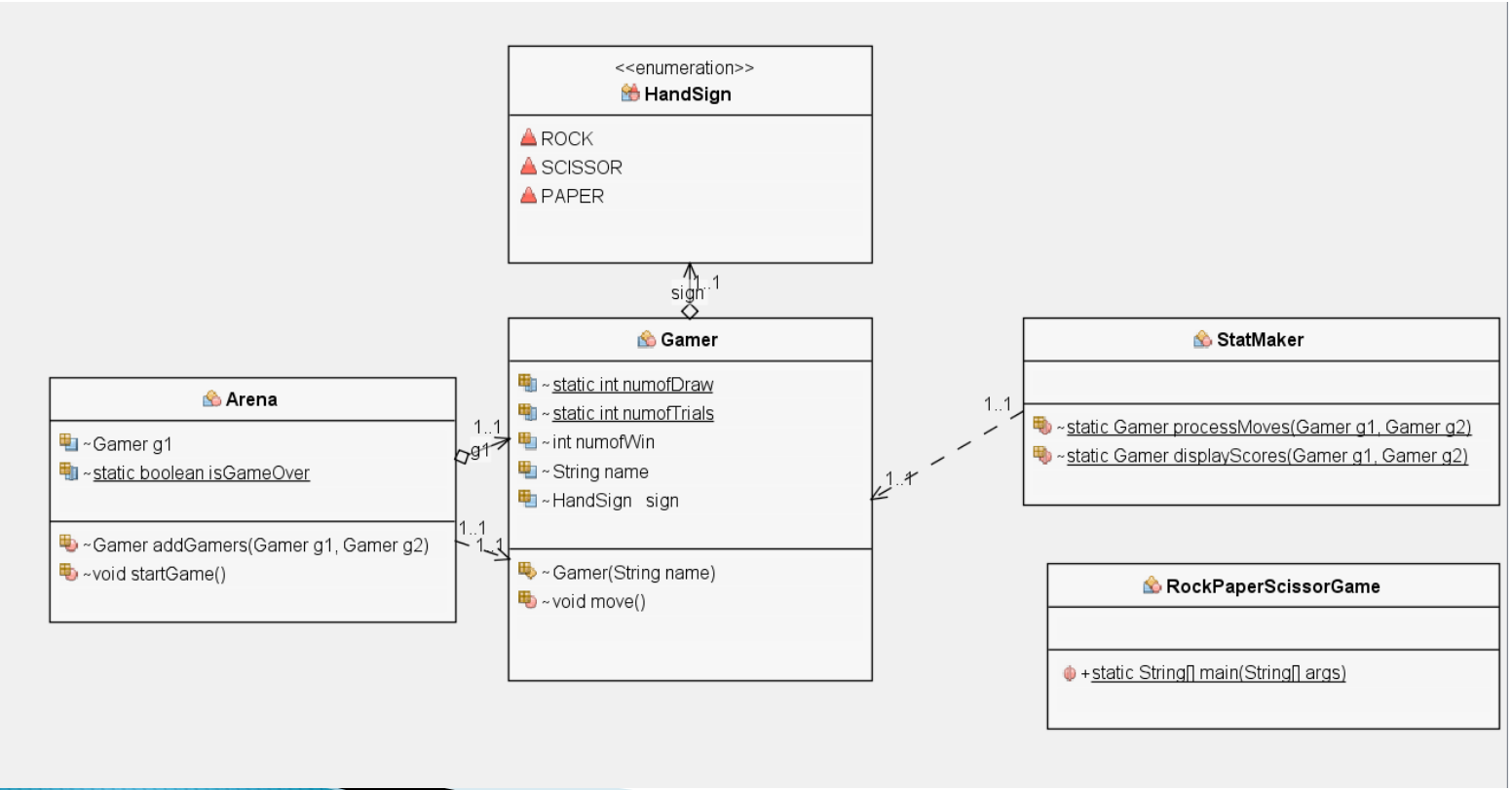
Rock Scissor Paper Game

Game Rules

- If the pair is player1 is paper and player2 is rock then player1 wins
- If the pair is player1 is scissors and player2 is paper then player1 wins
- If the pair is player1 is rock and player2 is scissors then player1 wins
- If Player1 is same as player2, then draw!
- Else player2 wins

Rock Scissor Paper Game

What we need are…



Game Logic

RockScissorPaper Class (main class)

```
public static void main(String[] args) {

    Gamer g1 = new Gamer("gamer 1");
    Gamer g2 = new Gamer("gamer 2");
    Arena arena = new Arena();
    arena.addGamers(g1,g2);
    arena.startGame();
}
```

Arena Class

```
void addGamers(Gamer g1, Gamer g2) {
    this.gamer1=g1;
    this.gamer2=g2;
}
```

```
void startGame() {
    while(!isGameOver)
    {
    gamer1.move();
    gamer2.move();
    StatMaker.processMoves(gamer1,gamer2);
    StatMaker.displayScores(gamer1,gamer2);
    }
}
```

Gamer Class

Game Logic

```
void move() {
    boolean isInputValid=false;
    Scanner input = new Scanner(System.in);
    System.out.println(this.name +" please enter p: paper s:scissor: r:rock q:quit");

    do{
        char inChar = input.next().toLowerCase().charAt(0);
          switch(inChar)
            {
                case 'q':
                    Arena.isGameOver=true;
                    break;
              case 'p':
                    sign=HandSign.PAPER;
                    break;
              case 'r':
                    sign=HandSign.ROCK;
                    break;
              case 's':
                    sign=HandSign.SCISSOR;
                    break;
              default:
                  System.out.println("your input is invalid. Please try again");
                  isInputValid=true;
                  break;
            }
    }
    while(isInputValid);
}
```

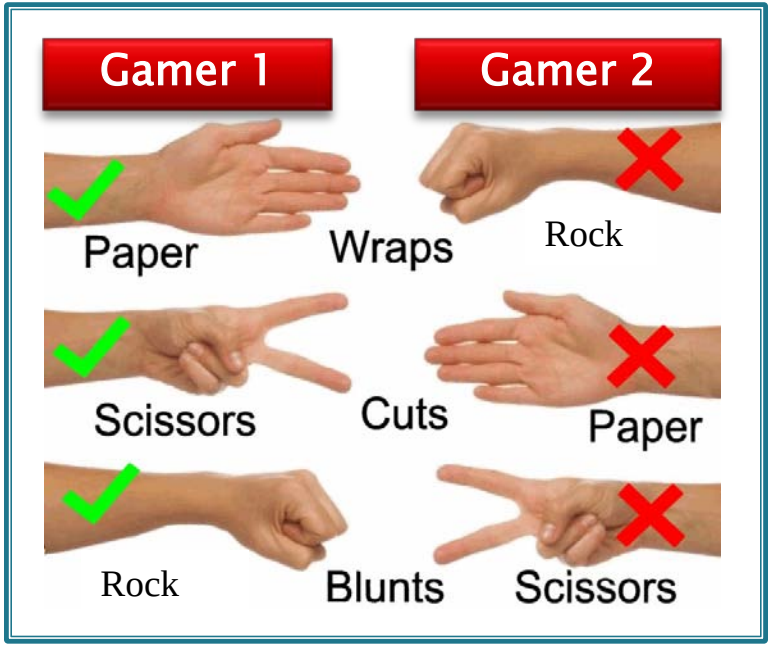
Rock Scissor Paper Game

StatMaker Class

```
static void processMoves(Gamer g1, Gamer g2) {
    if (g1.sign==null || g2.sign==null || Arena.isGameOver) return;
    else if(g1.sign == g2.sign)
        Gamer.numofDraw++;
    else if(g1.sign==HandSign.PAPER && g2.sign == HandSign.ROCK)
        g1.numofWin++;
            else if(g1.sign==HandSign.SCISSOR && g2.sign == HandSign.PAPER)
        g1.numofWin++;
            else if(g1.sign==HandSign.ROCK && g2.sign == HandSign.SCISSOR)
        g1.numofWin++;
    else
        g2.numofWin++;

    Gamer.numofTrials++;

}
```




```
static void displayScores(Gamer g1, Gamer g2) {
    System.out.println(g1.name+ " : "+g1.numofWin + " %" + (Gamer.numofTrials==0 ? 0: (float)g1.numofWin/Gamer.numofTrials)*100f);
     System.out.println(g2.name+ " : "+g2.numofWin + " %" + (Gamer.numofTrials==0 ? 0: (float)g2.numofWin/Gamer.numofTrials)*100f);
     System.out.println("Num of draws: " + Gamer.numofDraw);
}
```

Let's make this game.

Thanks ☺

Lab exercise : Calling a Constructor From a Constructor

- Create a class called Person where, id, firstName, lastName, and age attributes are declared.
- Create four constructors
  - In first, set all attributes;
  - In the second, assign only first and last names, automatically increment id and age will be set to zero;
  - In the third, only set id, and set rest to default values;
  - Int the last only increment id, set the rest to default values.