目录

Day11. Java

1 java.util.Date

- 封装一个毫秒值，表示一个具体的时间点
- 毫秒值：从1970-1-1 0点开始的毫秒值
- 创建对象
  - new Date()
    封装系统当前时间毫秒值
  - new Date(900000000000L)
    封装指定的毫秒值
- 方法
  - getTime()
  - setTime(long t)
    存取内部的毫秒值
  - compteTo(Date d)
    当前对象，与参数对象d，比较大小
    当前对象大，正数
    当前对象小，负数
    相同，0

2 java.text.SimpleDateFormat

- 日期格式工具
- Date对象，格式化成字符串
- 日期字符串，解析成Date对象
- 创建对象
  - new SimpleDateFormat(格式)
    格式： 参考 API 文档
    "yyyy-MM-dd HH:mm:ss"
    "dd/MM/yyyy"
    "MM/dd/yyyy"
    "HH:mm:ss"
    "yy-M-d H:m"
    "yyyy年MM月dd日"

- 方法
  - format(Date)
    Date对象格式化成日期字符串

  - parse(String)
    日期字符串，解析成Date对象

Date

项目：day1101_Date
类：day1101.Test1

```
package day1101;

import java.text.SimpleDateFormat;
import java.util.Date;

public class Test1 {
   public static void main(String[] args) {
      Date a = new Date();
      Date b = new Date(900000000000L);
      //println()方法内部会
      //调用对象的toString()获取字符串
      //再打印这个字符串
      System.out.println(a);
      System.out.println(b);
      System.out.println(a.getTime());
      System.out.println(b.getTime());
      a.setTime(0);
      System.out.println(a);
      System.out.println(a.compareTo(b));

      SimpleDateFormat sdf =
```

```
         new SimpleDateFormat(
         "yyyy-MM-dd HH:mm:ss");

      String s1 = sdf.format(a);
      String s2 = sdf.format(b);
      System.out.println(s1);
      System.out.println(s2);
   }
}
```

Test2

```
package day1101;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class Test2 {
   public static void main(String[] args) throws ParseException {
      /*
       * 输入生日,
       * 显示: 您已经生存了 xxx 天
       *
       * "1994-9-12"  解析成 Date对象
       */
      System.out.print("输入生日(yyyy-MM-dd): ");
      String s = new Scanner(System.in).nextLine();

      SimpleDateFormat sdf =
       new SimpleDateFormat("yyyy-MM-dd");
      //ctrl+1, 选择 add throws...
      Date d = sdf.parse(s);
      long t =
       System.currentTimeMillis()-d.getTime();

      t = t/1000/60/60/24;
      System.out.println("您已经生存了"+t+"天");
   }
}
```

3 集合

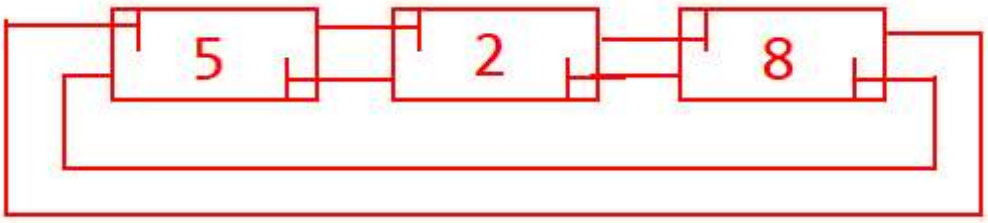
- 用来存放一组数据的数据结构
- 数组的缺点
  - 长度不可变
  - 执行复杂操作，操作繁琐
  - 访问方式单一，只能用下标访问

3.1 集合继承结构

```
Collection 接口
   |- List 接口
         |- ArrayList
         |- LinkedList
   |- Set 接口
         |- HashSet
         |- TreeSet

Map 接口
   |- HashMap
   |- TreeMap

Iterator 接口
Collections 工具类
```

4 LinkedList

- 双向链表
- 两端效率高



- 方法
  - add(数据)
    添加数据
  - get(int i)
    用下标访问数据
  - remove(int i)
    用下标移除数据，并返回被移除的值

- remove(数据)
  找到第一个相等数据，移除
  返回布尔值，表示是否找到并删除数据

- size()
  数据的数量

- iterator()
  用来辅助创建迭代器对象

- addFirst(),addLast()
- getFirst(),getLast()
- removeFirst(),removeLast()

LinkedList

项目：day1102_LinkedList
类：day1102.Test1

```
package day1102;

import java.util.Iterator;
import java.util.LinkedList;

public class Test1 {
   public static void main(String[] args) {
      /*
       * <> 泛型
       *    限制集合中，存放的数据类型
       *    泛型和集合，不支持基本类型
       */
      LinkedList<String> list =
            new LinkedList<>();
      list.add("aaa");
      list.add("ggg");
      list.add("ttt");
      list.add("ooo");
      list.add("qqq");
      list.add("ccc");
      list.add("fff");
      list.add("aaa");
      list.add("aaa");
      System.out.println(list.size());//数据的数量，元素数量
      System.out.println(list);
      System.out.println(list.get(0));//访问0位置数据
          System.out.println(list.get(list.size()-1));    // 访 问 末 尾 位 置 数
据
      System.out.println(list.remove(4));//删除下标4位置数据
      System.out.println(list);
      System.out.println(list.remove("aaa"));//删除第一个相等数据
      System.out.println(list);
```

```
        //双向链表，下标遍历效率低
        for(int i=0;i<list.size();i++) {
           String s = list.get(i);
           System.out.println(s);
        }

        //双向链表，迭代器遍历效率高
        //新建迭代器对象
        Iterator<String> it = list.iterator();
        //当还有下一个
        while(it.hasNext()) {
           String s = it.next();
           System.out.println(s);
        }



    }
}
```

Test2

```
package day1102;

import java.util.Iterator;
import java.util.LinkedList;

public class Test2 {
    public static void main(String[] args) {
        /*
         * 添加100万数据量
         * 测试下标遍历和迭代器遍历的效率
         */
        LinkedList<Integer> list = new LinkedList<>();
        for(int i=0;i<1000000;i++) {
           list.add(1);//list.add(Integer.valueOf(1));
        }

        System.out.println("数据量: "+list.size());

        System.out.println("\n--下标遍历-----------");
        long t = System.currentTimeMillis();
        f1(list);
        System.out.println(
         System.currentTimeMillis()-t);

        System.out.println("\n--迭代遍历-----------");
        t = System.currentTimeMillis();
        f2(list);
        System.out.println(
         System.currentTimeMillis()-t);
```

```
    }

    private static void f1(LinkedList<Integer> list) {
        for(int i=0;i<list.size();i++) {
            list.get(i);
        }
    }

    private static void f2(LinkedList<Integer> list) {
        Iterator<Integer> it = list.iterator();
        while(it.hasNext()) {
            it.next();
        }
    }
}
```

丑数

能被2,3,5整除多次，整除成1
2,3,4,5,6,8,9,10,12,15,16,18,20...

求第n个丑数

项目：day1103_丑数
类：day1103.Test1

```
package day1103;

import java.util.LinkedList;
import java.util.Scanner;

public class Test1 {
    public static void main(String[] args) {
        System.out.print("求第几个丑数：");
        int n = new Scanner(System.in).nextInt();
        long r = f(n);
        System.out.println(r);
    }

    private static long f(int n) {
        /* xx xx xx xx xx xx
         * ----------------------------------------
         *                    i
         * 6 xx xx
         * -----------------
         *         j
         * 10
```

```
         * ----------------
         *   k
         *
         * 2 3 4 5
         */
        LinkedList<Long> list2 = new LinkedList<>();
        LinkedList<Long> list3 = new LinkedList<>();
        LinkedList<Long> list5 = new LinkedList<>();
        list2.add(2L);
        list3.add(3L);
        list5.add(5L);

        long r = 0;//用来存放每一次取出的丑数
        //从第1个，求到第n个
        for(int i=1;i<=n;i++) {
            //第一步：移除最小值
            long a = list2.getFirst();
            long b = list3.getFirst();
            long c = list5.getFirst();
            r = Math.min(a, Math.min(b, c));
            if(r == a) list2.removeFirst();
            if(r == b) list3.removeFirst();
            if(r == c) list5.removeFirst();
            //第二步：分别乘2,3,5，放入集合
            list2.add(r*2);
            list3.add(r*3);
            list5.add(r*5);
        }
        return r;
    }
}
```

5 java.util.ArrayList

- 内部使用数组存放数据
- 增删数据，效率可能降低
- 访问任意位置效率高

- 内部数组，默认初始容量 10
- 存满后，新建1.5倍容量新数组

- 创建对象
  - new ArrayList()
    初始容量 10

  - new ArrayList(1000)
    初始容量 1000

- 方法
  - 与双向链表相同
  - 没有两端操作数据的方法

猜游戏

猜过的，如果重复猜，向用户提示
添加次数显示

```
猜（第1次）：ABCDE
猜（第2次）：ABCDE
猜（第3次）：ABCDE
50已经才过了
猜（第3次）：
```

day1003_猜游戏
复制
day1104_猜游戏

```
package day1104;

import java.util.ArrayList;
import java.util.Scanner;

public abstract class GuessGame {
   public void start() {
      //两种游戏的通用流程
      //产生随机值
      String r = suiJi();
      //显示提示
      tiShi();
      ArrayList<String> list = new ArrayList<>();
      while(true) {
         System.out.print("猜(第 "+(list.size()+1)+" 次): ");
         String c = new Scanner(System.in).nextLine();
         //判断是否猜过
         if(list.contains(c)) {
            System.out.println(c+"已经猜过了");
            continue;
         }
         list.add(c);

         //比较c和r，并得到比较的结果
         String result = biJiao(c, r);
         System.out.println(result);
         //result是否是猜对的结果
         if(caiDui(result)) {
            break;
         }
```

```
        }

    }

    public abstract String suiJi();
    public abstract void tiShi();
    public abstract String biJiao(String c, String r);
    public abstract boolean caiDui(String result);
}
```

6 飞机大战-敌人出现

- 每30帧随机出现一个敌人
  - 60%，小飞机
  - 30%，大飞机
  - 10%，小蜜蜂
- 用ArrayList存放敌人
- 绘制敌人，遍历集合，绘制所有敌人
- 移动一帧，遍历集合，每个敌人移动一帧
- 敌人中添加判断走出画面的方法
- 当一个敌人移动了一帧，判断走出画面，从集合删除

day0903_飞机大战
复制
day1105_飞机大战

Enemy

```
package day0804;

import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.util.Random;

public abstract class Enemy {
    //初始值，要在子类构造方法中赋值
    BufferedImage[] imgs;
    BufferedImage img;
    int x;
    int y;
```

```
    public void paint(Graphics g) {
        g.drawImage(img,x,y,null);
    }

    public abstract void step();

    public int rndX() {
        // [0, 400-图片宽度)
        return new Random().nextInt(400-img.getWidth());
    }

    public boolean isOut() {
        return y>654 || x<-img.getWidth() || x>400;
    }
}
```

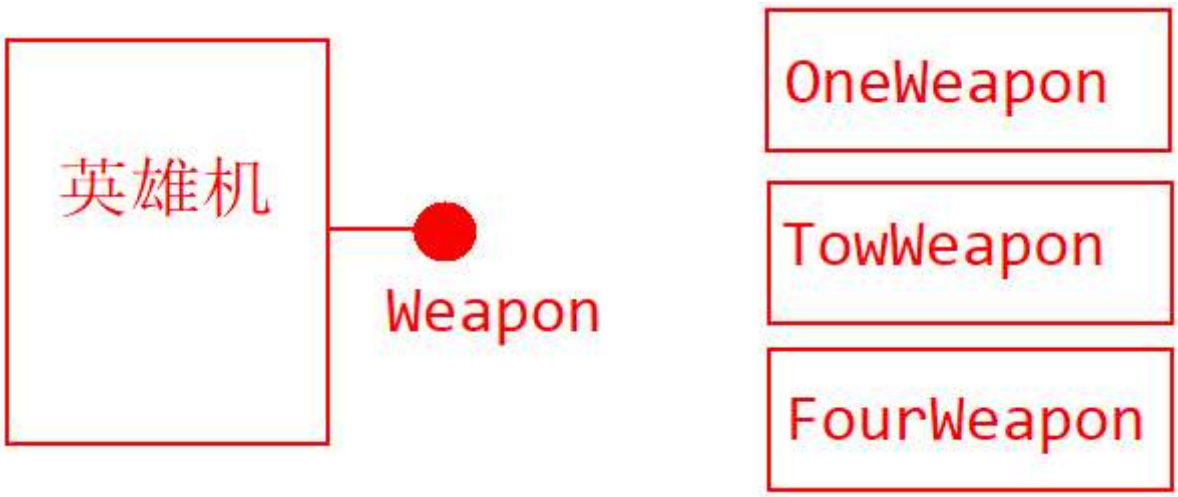
GamePanel

```
package day0804;

import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.JPanel;

public class GamePanel extends JPanel {
    Background bg = new Background();
    Hero hero = new Hero();
    ArrayList<Enemy> enemys = new ArrayList<>();
    //帧计数
    int count;

    public GamePanel() {
        //设置面板的期望大小
        setPreferredSize(new Dimension(400, 654));
    }

    /*
     * 固定的绘图方法
     * 由系统自动调用
     */
    @Override
    public void paint(Graphics g) {
        bg.paint(g);
        //绘制所有敌人
        paintEnemys(g);
        hero.paint(g);
    }
```

```
    private void paintEnemys(Graphics g) {
        Iterator<Enemy> it = enemys.iterator();
        while(it.hasNext()) {
            Enemy e = it.next();
            e.paint(g);
        }
    }

    //动起来方法
    public void action() {
        //设置鼠标监听器
        addMouseMotionListener(new MouseAdapter() {
            @Override
            public void mouseMoved(MouseEvent e) {
                hero.setTarget(e.getX(), e.getY());
            }
        });


        //画面一帧一帧的循环播放
        while(true) {
            count++;//帧计数增加

            bg.step();//背景移动

            enemysIn();//敌人出现
            enemysStep();//所有敌人移动一帧

            hero.step();

            //通知底层系统，重绘画面
            //系统受到通知后，会自动调用 paint() 方法
            repaint();
            //暂停 1/60 秒 (60 fps)
            try {
                Thread.sleep(1000/60);
            } catch (InterruptedException e) {
            }
        }
    }

    private void enemysIn() {
        //每30帧出现
        if(count%30 == 0) {
            double r = Math.random();
            if(r<0.6) {
                enemys.add(new Airplane());
            } else if(r<0.9) {
                enemys.add(new BigPlane());
            } else {
                enemys.add(new Bee());
            }
```

```
        }
    }

    private void enemysStep() {
        Iterator<Enemy> it = enemys.iterator();
        while(it.hasNext()) {
            Enemy e = it.next();
            e.step();
            if(e.isOut()) {
                /*
                 * 再迭代遍历期间,
                 * 不允许使用集合来增删数据
                 * 如果要删除数据，必须使用迭代器的删除方法
                 *
                 */
                //enemys.remove(e);
                it.remove();//删除刚刚取出的数据
            }
        }
    }
}
```

7 飞机大战-发射子弹



day1105_飞机大战
复制
day1106_飞机大战

8 作业

- 重写
  - day1103_丑数
- 复习 LinkedList和ArrayList
- 输入春节日期，显示距离春节还有xxx天