```java
package game;

//TODO We need to make sure when we throw the exception for a balance when it goes below zero, that we then
// create a debtobligation for a player for the amount that he owes.

public class Account
{
    @SuppressWarnings("serial")
    public class IllegalAmountException extends Exception {
        public IllegalAmountException(String string) {
            super(string);
        }
    }
    @SuppressWarnings("serial")
    public class InsufficientFundsException extends Exception {
        public InsufficientFundsException(String string) {
            super(string);
        }


    }

    private int balance;
    private final static int DEFAULT_START_BALANCE = 30000;

    public Account() {
        this(DEFAULT_START_BALANCE);
    }

    public Account(int balance)
    {
        this.balance = balance;
    }

    public int getBalance()
    {
        return balance;
    }

    // checks that you're not attempting to set the score below zero
    // then sets the score to the given integer
    public void setBalance(int balance) throws Exception
    {
        // check for negative total
        if (balance < 0)
            throw new Exception("Account balance may not be negative");
        this.balance = balance;
    }

```

```
50      // adds an amount to the score, provided that its does not bring the
    score
51      // below zero, or above intMax
52      public int deposit(int amount) throws IllegalAmountException
53      {
54          // refuse sub-zero numbers
55          if (amount < 0)
56              throw new IllegalAmountException("May not deposit negative
    amount");
57          // test for integer overrun
58          if (balance > Integer.MAX_VALUE - amount) // Corrected from score +
    amount > Integer.MAX_VAlUE - would fail if amount + score exceeded
    INTEGER.MAXVALUE - which defies the purpose
59              throw new IllegalAmountException(
60                      "Cannot deposit amount - balance will exceed
    Integer.MAX_VALUE");
61          balance = balance + amount;
62          return balance; // returns new score
63      }
64
65      // withdraws an amount from the score, refuses negative numbers, and
    refuses
66      // to withdraw more than there is in score.
67      public int withdraw(int amount) throws InsufficientFundsException,
    IllegalAmountException
68      {
69          //refuse negative numbers
70          if (amount < 0)
71              throw new IllegalAmountException("May not withdraw negative
    amount");
72          //refuse to withdraw more than total
73          if (amount > this.balance)
74              throw new InsufficientFundsException("May not withdraw more than
    balance");
75          this.balance = this.balance - amount;
76          return balance; // returns new balance
77      }
78
79      @Override
80      public String toString()
81      {
82          return "Account [balance = " + balance + "]";
83      }
84
85  }
86
```