UNIVERSITI MALAYA

FAKULTI SAINS KOMPUTER DAN TEKNOLOGI MAKLUMAT
Faculty of Computer Science and Information Technology

LAB REPORT

VIVA 3

REPRESENT BY:
NEWBIE MEMERS

Name	Matric Number
TITUS TAN YU FAN	23004883
WONG HOONG LIANG	23053016
SIM JING JIA	23004919
LIM HONG YU	23004973
SOON MING HONG	23004950

Question 1

Problem Description

To defeat the BlackMage!

Need three class :

1. Phantom

-initializing Phantom's accuracy and critRate between 0.00 and 1.00

2. BlackMage

-initializing default values for hp(100), evasion(5%) and critResistance(10%)

3. Main class: TesterVIVA1

-calculate damage and hp of BlackMage

-return combat log

Solution

We created three class

1. Phantom.java

public Phantom(double accuracy, double critRate)

- Constructor with parameters initialize accuracy and critRate

public void setAccuracy(double newAccuracy) / setCrisRate(double newCritrate)

- Mutator method to set a new accuracy / critRate value, with a limit between 0.00 and 1.00

public double getAccuracy() / getCrisrate()

- Accessor method to retrieve the current accuracy value / critRate value

2. BlackMage.java

public BlackMage()

- Constructor with default values for hp, critResistance, and evasion

public double getCrisResistance() / getEvasion()

- Accessor method to retrieve the current critResistance value / evasion value

public void setCrisRate(double critResistance) / setEvasion(double evasion)

- Mutator method to set a new critResistance value / new evasion value

public int getHP()

- Accessor method to retrieve the current hp value

public void setHP(int hp)

- Mutator method to set a new hp value

3. TesterVIVA1.java

Main Method (main):

- The main method creates an instance of the TesterVIVA1 class (tester) and prints the result of the toString method.

Damage Calculation Method (damage):

- This method calculates the damage inflicted by the Phantom to the BlackMage based on accuracy, evasion, and critical hit resistance.
- We use random values to simulate the chance of a critical hit, a normal hit, or a missed hit.

Simulation Method (toString):

- This method simulates a combat scenario between the Phantom and the BlackMage.
- We repeatedly call the damage method until the Black Mage's health (hp) reaches 0.
- It generates a combat log (result) indicating each hit, including whether it was a miss, a normal hit, or a critical hit.
- The combat log is constructed using a StringBuilder and returned as a formatted string.

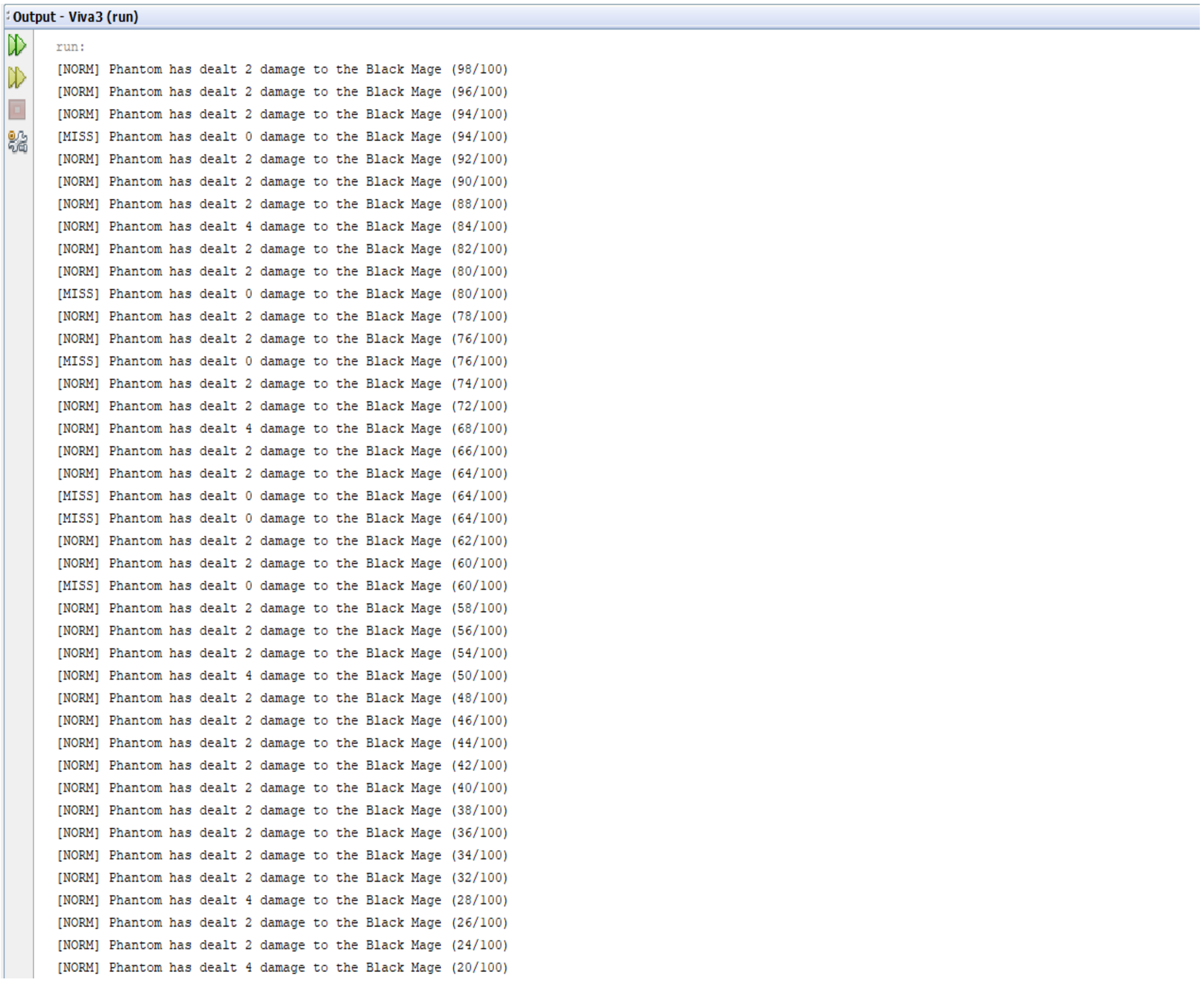
So…, at last the BlackMage will be defeated by our HERO Phantom!!!

```java
public class Phantom {
    private double accuracy;
    private double critRate;

    public Phantom() {
        this.accuracy = 1.00;
        this.critRate = 1.00;
    }

    public Phantom(double accuracy, double critRate) {
        this.accuracy = accuracy;
        this.critRate = critRate;
    }

    public void setAccuracy(double newAccuracy) {
        if (newAccuracy >= 0.00 && newAccuracy <= 1.00) {
            this.accuracy = newAccuracy;
        }else {
            System.out.println("Invalid accuracy. Please change the
accuracy to double between 0.00 to 1.00");
        }
    }

    public double getAccuracy() {
        return accuracy;
    }

    public void setCrisRate(double newCritRate) {
        if (newCritRate >= 0.0 && newCritRate <= 1.0) {
            this.critRate = newCritRate;
        } else {
            System.out.println("Invalid critRate. Please change the
crisRate to double between 0.00 to 1.00");
        }
    }

    public double getCrisrate() {
        return critRate;
    }
}
```

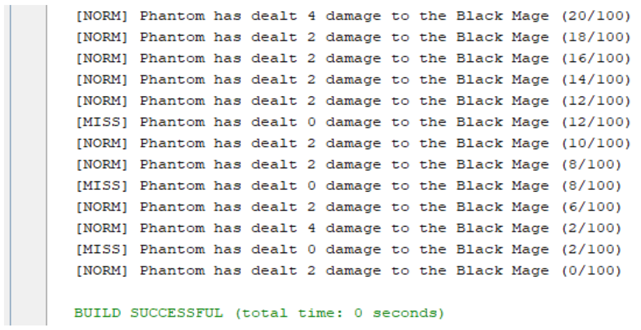
```java
public class BlackMage {
    private int hp;
    private double critResistance;
    private double evasion;


    public BlackMage () {
        hp = 100;
        critResistance = 0.1;
        evasion = 0.05;
    }

    public double getCrisResistance() {
        return critResistance;
    }
    public double getEvasion() {
        return evasion;
    }
    public int getHP() {
        return hp;
    }
}
```

```java
import java.util.Random;

public class TesterVIVA1 {
    public static void main(String[] args) {
        TesterVIVA1 tester = new TesterVIVA1();
        System.out.println(tester.toString());

    }

    protected static int damage(BlackMage blackMage, Phantom phantom) {
        Random rd = new Random();
        double totalEffectiveAccuracy = (phantom.getAccuracy()) * (1 -
(blackMage.getEvasion()));
        double totalEffectiveCritRate = (phantom.getCrisrate()) -
(blackMage.getCrisResistance());
        double accuracyRandom = rd.nextDouble();
        double critRandom = rd.nextDouble();

        if (accuracyRandom <= totalEffectiveAccuracy) {
            if (critRandom <= totalEffectiveCritRate) {
                return 4;
            } else {
                return 2;
            }
        }else {
            return 0;
        }
    }

    public String toString() {
        BlackMage blackMage = new BlackMage();
        Phantom p = new Phantom();
        p.setAccuracy(Math.random());
        p.setCrisRate(Math.random());
        Phantom phantom = new Phantom(p.getAccuracy(), p.getCrisrate());
        int hp = blackMage.getHP();
        StringBuilder result = new StringBuilder();
        while (hp > 0) {
            int damage = damage(blackMage, phantom);
            if (damage == 0) {
                result.append("[MISS] Phantom has dealt
").append(damage).append(" damage to the Black Mage (").append(hp-
damage).append("/100)\n");
            }
            else if ((hp - damage) < 0) {
                result.append("[NORM] Phantom has dealt
").append(damage).append(" damage to the Black Mage (0/100)\n");
            } else {
                result.append("[NORM] Phantom has dealt
").append(damage).append(" damage to the Black Mage (")
                        .append(hp - damage).append("/100)\n");
            }
            hp -= damage;
        }

        return result.toString();
    }
}
```

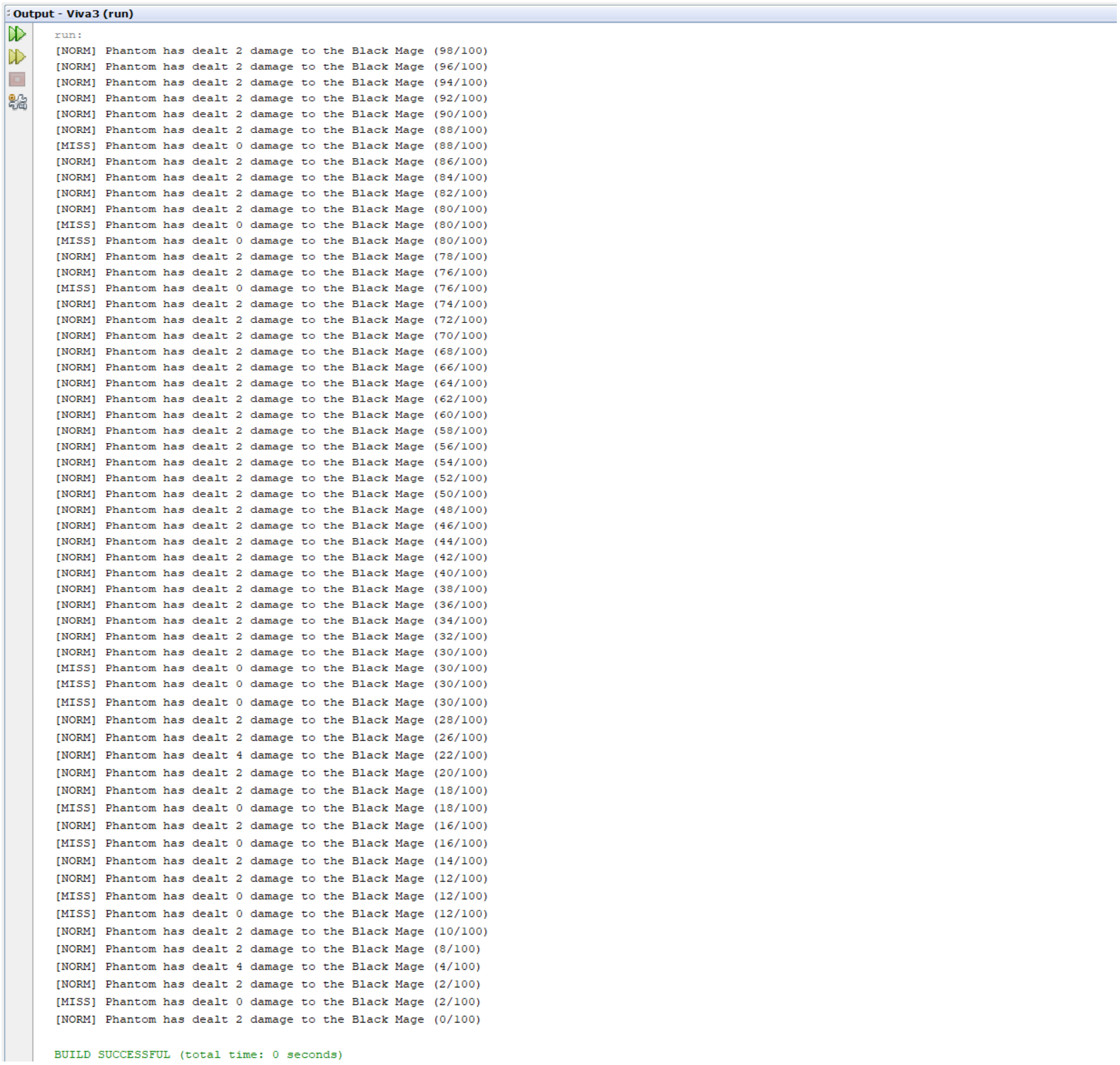
Sample Output


Output - Viva3 (run)

```
run:
[NORM] Phantom has dealt 2 damage to the Black Mage (98/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (96/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (94/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (94/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (92/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (90/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (88/100)
[NORM] Phantom has dealt 4 damage to the Black Mage (84/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (82/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (80/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (80/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (78/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (76/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (76/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (74/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (72/100)
[NORM] Phantom has dealt 4 damage to the Black Mage (68/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (66/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (64/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (64/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (64/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (62/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (60/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (60/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (58/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (56/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (54/100)
[NORM] Phantom has dealt 4 damage to the Black Mage (50/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (48/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (46/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (44/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (42/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (40/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (38/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (36/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (34/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (32/100)
[NORM] Phantom has dealt 4 damage to the Black Mage (28/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (26/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (24/100)
[NORM] Phantom has dealt 4 damage to the Black Mage (20/100)
```



```
[NORM] Phantom has dealt 4 damage to the Black Mage (20/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (18/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (16/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (14/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (12/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (12/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (10/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (8/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (8/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (6/100)
[NORM] Phantom has dealt 4 damage to the Black Mage (2/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (2/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (0/100)

BUILD SUCCESSFUL (total time: 0 seconds)
```



Output - Viva3 (run)

```
run:
[NORM] Phantom has dealt 2 damage to the Black Mage (98/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (96/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (94/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (92/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (90/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (88/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (88/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (86/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (84/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (82/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (80/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (80/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (80/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (78/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (76/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (76/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (74/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (72/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (70/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (68/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (66/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (64/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (62/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (60/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (58/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (56/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (54/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (52/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (50/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (48/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (46/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (44/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (42/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (40/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (38/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (36/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (34/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (32/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (30/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (30/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (30/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (30/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (28/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (26/100)
[NORM] Phantom has dealt 4 damage to the Black Mage (22/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (20/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (18/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (18/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (16/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (16/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (14/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (12/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (12/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (12/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (10/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (8/100)
[NORM] Phantom has dealt 4 damage to the Black Mage (4/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (2/100)
[MISS] Phantom has dealt 0 damage to the Black Mage (2/100)
[NORM] Phantom has dealt 2 damage to the Black Mage (0/100)

BUILD SUCCESSFUL (total time: 0 seconds)
```

Question 2

Problem description:

To design a sophisticated library system with two classes: Book and Library.

1. Book Class

- Attributes: title, author, ISBN
- Required: constructor, accessors, mutators

2. Library Class

- Find books by title or author.
- Add or remove books from the collection.
- Sort books by title.
- Special Rule: Books with ISBN ending in prime digits for the last two digits can't be borrowed.

Solution:

We need to create Book class and Library class as well as one main class, which is tester2.

1. Book Class:

- This class represents a book and includes attributes like ‘bookName’, ‘author’, and ‘ISBN’.
- It has a constructor to initialize these attributes and getter methods to retrieve them.

2. Library Class:

- This class represents a magical library.
- It contains multiple ‘HashMap’ collections to store book details such as ‘bookNameList’, ‘authorList’, ‘ISBNList’, etc.
- Functions include:
  - addBook(Book book): Adds a book to the library by storing its details in the HashMaps.
  - findBooksByAuthor(String input): Searches for books by author and displays their details.
  - findBooksByTitle(String input): Searches for books by title and displays their details.
  - borrowBook(String input): Checks if a book can be borrowed based on its ISBN number (with special rules for prime digits at the end).
  - sortBooks(): Sorts the books by title and displays their details.

- removeBook(String input): Removes a book from the library based on its ISBN number.
- displayLibrary(): Displays all books in the library sorted by title.

3. tester2 Class:

- This class contains the main method that acts as a test scenario for the library functionality.
- It creates a ‘hogwartsLibrary’ object and several ‘Book’ objects.
- It demonstrates adding books, searching for books by author and title, attempting to borrow a book, sorting books, removing a book, and displaying the updated library contents.

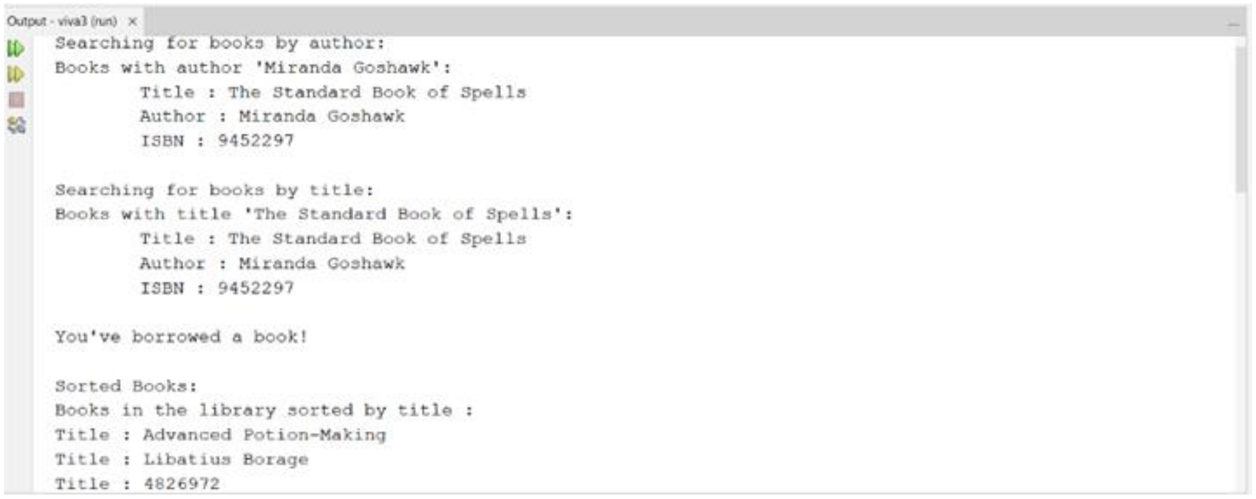
The code showcases the implementation of a simple library system for Hogwarts using Java classes. It demonstrates various operations one can perform in the library, such as adding, searching, borrowing, sorting, and removing books.

Source code:

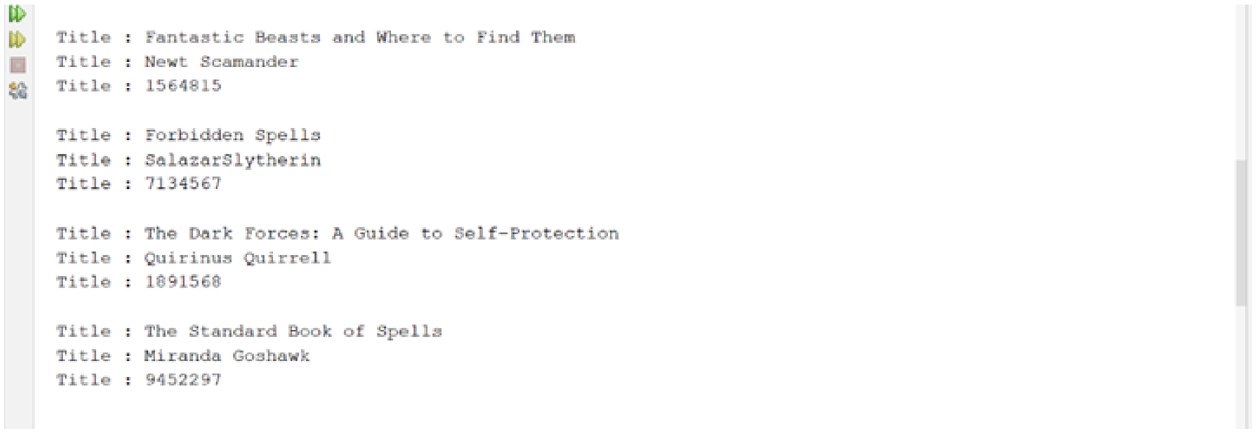
```java
public class Book {
    private String bookName;
    private String author;
    private String ISBN;

    public Book (String bookName, String author, String ISBN) {
        this.bookName = bookName;
        this.author = author;
        this.ISBN = ISBN;

    }

    public String getBookName() {
        return bookName;
    }

    public String getAuthor() {
        return author;
    }

    public String getISBN() {
        return ISBN;
    }
}



import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Library {
    private static int i;
    Map<String, Integer> bookNameList = new HashMap<>();
    Map<String, Integer> authorList = new HashMap<>();
    Map<Integer, String> ISBNList = new HashMap<>();
    Map<Integer, String> bookNameSearch = new HashMap<>();
    Map<Integer, String> authorSearch = new HashMap<>();
    Map<String, Integer> ISBNAddNRemove = new HashMap<>();
    private static List<Integer> IntegerValue = new ArrayList<>();
    private static List<Integer> borrowedBooks = new ArrayList<>();


    protected void addBook(Book book) {
        bookNameList.put(book.getBookName(), i);
        authorList.put(book.getAuthor(), i);
        ISBNList.put(i, book.getISBN());
        bookNameSearch.put(i, book.getBookName());
        authorSearch.put(i, book.getAuthor());
        ISBNAddNRemove.put(book.getISBN(), i);
        i++;

    }

    protected void findBooksByAuthor(String input) {
        int authorKey = authorList.get(input);
        System.out.printf("Books with author '%s':\n", input);
        System.out.println("\tTitle : " + bookNameSearch.get(authorKey));
        System.out.println("\tAuthor : " + input);
        System.out.println("\tISBN : " + ISBNList.get(authorKey));

    }

    protected void findBooksByTitle(String input) {
        int titleKey = bookNameList.get(input);
        System.out.printf("Books with title '%s':\n", input);
        System.out.println("\tTitle : " + input);
        System.out.println("\tAuthor : " + authorSearch.get(titleKey));
        System.out.println("\tISBN : " + ISBNList.get(titleKey));

    }

    public boolean borrowBook(String input) {
        int intInput = 0;
        try {
            intInput = Integer.parseInt(input);
        } catch (NumberFormatException e) {
            System.out.println("Number Format Exception: " +
e.getMessage());
            return false;
        }
        if (isValidISBN(intInput)) {
            borrowedBooks.add(ISBNAddNRemove.get(input));
            return true;
        } else {
            return false;
        }
    }

    protected void sortBooks() {
        List<Map.Entry<String, Integer>> entryList = new
ArrayList<>(bookNameList.entrySet());
        Collections.sort(entryList,
Comparator.comparing(Map.Entry::getKey));

        for (Map.Entry<String, Integer> entry : entryList) {
            IntegerValue.add(entry.getValue());
        }
        System.out.println("Books in the library sorted by title : ");
        for (int i = 0; i < IntegerValue.size(); i++) {
            System.out.println("Title : " +
bookNameSearch.get(IntegerValue.get(i)));
            System.out.println("Title : " +
authorSearch.get(IntegerValue.get(i)));
            System.out.println("Title : " +
ISBNList.get(IntegerValue.get(i)));
            System.out.println();
        }
    }

    protected void removeBook(String input) {
        int IntegerToBeRemoved = ISBNAddNRemove.get(input);
        String bookNameRemoval = bookNameSearch.get(IntegerToBeRemoved);
        String authorRemoval = authorSearch.get( IntegerToBeRemoved);

        if (IntegerValue.contains(IntegerToBeRemoved)) {
            bookNameSearch.remove(IntegerToBeRemoved);
            authorSearch.remove(IntegerToBeRemoved);
            ISBNList.remove(IntegerToBeRemoved);
            bookNameList.remove(bookNameRemoval);
            authorList.remove(authorRemoval);
            IntegerValue.remove(IntegerValue.indexOf(IntegerToBeRemoved));
        } else {
            System.out.println("Book not found in the library.");
        }
    }

    protected void displayLibrary() {
        System.out.println("Books in the library sorted by title : ");
        for (Map.Entry<String, Integer> entry : bookNameList.entrySet()) {
            int i = entry.getValue();
            if (!borrowedBooks.contains(i)) {
                System.out.println("Title : " + bookNameSearch.get(i));
                System.out.println("Author : " + authorSearch.get(i));
                System.out.println("ISBN : " + ISBNList.get(i));
                System.out.println();
            }
        }
    }
    private boolean isValidISBN(int intInput) {
        int lastOne = intInput % 10;
        intInput /= 10;
        int lastSecond = intInput % 10;
        int[] array = {2, 3, 5, 7};
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                if ((lastOne == array[i]) && (lastSecond == array[j])) {
                    return false;
                }
            }
        }
        return true;
    }
}




public class tester2 {
    public static void main(String[] args) {
        Library hogwartsLibrary = new Library();
        Book book1 = new Book("The Standard Book of Spells", "Miranda
Goshawk", "9452297");
        Book book2 = new Book("Advanced Potion-Making", "Libatius Borage",
"4826972");
        Book book3 = new Book("Fantastic Beasts and Where to Find Them",
"Newt Scamander", "1564815");
        Book book4 = new Book("The Dark Forces: A Guide to Self-
Protection", "Quirinus Quirrell", "1891568");
        Book book5 = new Book("Forbidden Spells", "Salazar Slytherin",
"7134567");

        // ISBN ends in prime digits
        hogwartsLibrary.addBook(book1);
        hogwartsLibrary.addBook(book2);
```

```java
        hogwartsLibrary.addBook(book3);
        hogwartsLibrary.addBook(book4);
        hogwartsLibrary.addBook(book5);
        System.out.println("Searching for books by author:");
        hogwartsLibrary.findBooksByAuthor("Miranda Goshawk");
        System.out.println("\nSearching for books by title:");
        hogwartsLibrary.findBooksByTitle("The Standard Book of Spells");
        boolean canBorrow = hogwartsLibrary.borrowBook("9452297");
        if (canBorrow) {
            System.out.println("\nYou've borrowed a book!");
        } else {
            System.out.println("\nSorry, this book is not available for
borrowing.");
        }
        System.out.println("\nSorted Books:");
        hogwartsLibrary.sortBooks();
        hogwartsLibrary.removeBook("1564815");
        // Fantastic Beasts and Where to Find Them
        System.out.println("\nUpdated Library:");
        hogwartsLibrary.displayLibrary();
    }
}
```

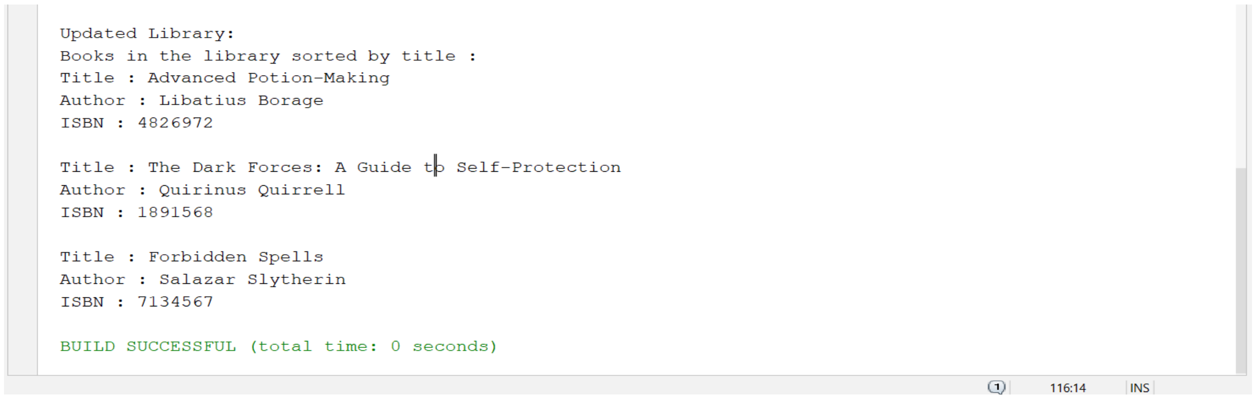
Sample Input & Output:


Output - viva3 (run)

```
Searching for books by author:
Books with author 'Miranda Goshawk':
        Title : The Standard Book of Spells
        Author : Miranda Goshawk
        ISBN : 9452297

Searching for books by title:
Books with title 'The Standard Book of Spells':
        Title : The Standard Book of Spells
        Author : Miranda Goshawk
        ISBN : 9452297

You've borrowed a book!

Sorted Books:
Books in the library sorted by title :
Title : Advanced Potion-Making
Title : Libatius Borage
Title : 4826972
```



```
Title : Fantastic Beasts and Where to Find Them
Title : Newt Scamander
Title : 1564815

Title : Forbidden Spells
Title : SalazarSlytherin
Title : 7134567

Title : The Dark Forces: A Guide to Self-Protection
Title : Quirinus Quirrell
Title : 1891568

Title : The Standard Book of Spells
Title : Miranda Goshawk
Title : 9452297
```



```
Updated Library:
Books in the library sorted by title :
Title : Advanced Potion-Making
Author : Libatius Borage
ISBN : 4826972

Title : The Dark Forces: A Guide to Self-Protection
Author : Quirinus Quirrell
ISBN : 1891568

Title : Forbidden Spells
Author : Salazar Slytherin
ISBN : 7134567

BUILD SUCCESSFUL (total time: 0 seconds)
```

116:14 INS