```java
public class SSD extends HardDrive {
    private int TeraBytesWritten;

    //accessors and mutators
    public int getTeraBytesWritten() {
        return TeraBytesWritten;
    }

    public void setTeraBytesWritten(int teraBytesWritten) {
        TeraBytesWritten = teraBytesWritten;
    }

    //constructor
    public SSD(String make, String model, int capacity, int teraBytesWritten) {
        super(make, model, capacity);
        this.TeraBytesWritten = 150;
    }
}

```

```java
import java.util.Random;

public class Hero extends Human{
    public Hero(String name) {
        super(name);
    }

    public void slashes(Zombie zombie) {
        //randomizes the damage done to the zombie
        Random random = new Random();
        int damageValue = random.nextInt(6) + 1;
        zombie.setHealth(zombie.getHealth() - damageValue);
        System.out.println(getName() + ": SLASH!");

        //if the zombie is still alive
        if (zombie.isAlive()) {
            System.out.println(zombie.getName() + " now has
   " + zombie.getHealth());
        //if the zombie dies, the hero wins!
        } else {
            System.out.println("I love the smell of
  necrosis in the morning!  Smells like...victory!");
        }
    }
}
```

```java
import java.util.Objects;

public class Fruit {

    private String name;
    private String color;

    //using accessors, mutators and constructors
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public Fruit(String name, String color) {
        this.name = name;
        this.color = color;
    }

    // copy constructor
    Fruit(Fruit c) {
        name = c.name;
        color = c.color;
    }

    // Overriding the toString of Object class
    @Override
    public String toString() {
        return name;
    }

    @Override
    public boolean equals(Object o) {
        if (o.getClass() != this.getClass()) {
            return false;
        } else {
```

```
47              Fruit fruit = (Fruit) o;
48              if (fruit.getName().equals(this.getName())){
49                  return true;
50              } else {
51                  return false;
52              }
53          }
54      }
55  }
56
```

```java
import java.util.Random;

public class Human {

    private String name;
    private int health;

    //accessors and mutators
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getHealth() {
        return health;
    }

    public void setHealth(int health) {
        this.health = health;
    }

    public boolean isAlive() {
        return getHealth() > 0;
    }

    //constructor
    public Human(String name) {
        this.name = name;
        this.health = 25; //default setting the health to 25
    }

    //creating the roll function that will determine if the zombie or the hero gets a turn
    public int roll() {
        return new Random().nextInt(6) + 1;
    }
}
```

```java
import java.util.Random;

public class Zombie extends Human{

    public Zombie(String name) {
        super(name);
    }

    public void bites (Hero hero) {
        //randomly setting a value for the damage that the hero got from the zombie
        Random random = new Random();
        int damageValue = random.nextInt(6) + 1;

        //the current health minus the damage that was taken
        hero.setHealth(hero.getHealth() - damageValue);
        System.out.println(getName() + ": CHOP!");

        //if the hero is still alive
        if (hero.isAlive()) {
            System.out.println(hero.getName() + " now has " + hero.getHealth());
        //if the hero dies, the zombie wins!
        } else {
            System.out.println("Mmmm! Delicious brains!");
        }
    }

}

```

```java
public class HardDrive {
    private String make;
    private String model;
    private int capacity;

    //Accessors and Mutators
    public String getMake() {
        return make;
    }

    public void setMake(String make) {
        this.make = make;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public int getCapacity() {
        return capacity;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    //Constructor
    public HardDrive(String make, String model, int capacity) {
        this.make = make;
        this.model = model;
        this.capacity = 1;
    }
}
```

```
 1 public class Assignment3a {
 2     public static void main(String[] args) {
 3         SSD SamsungT5 = new SSD("Samsung", "T5", 1, 150);
 4 
 5         System.out.println("HardDrive: " +
 6                            "\nMake: " + SamsungT5.getMake
   () +
 7                            "\nModel: " + SamsungT5.
   getModel() +
 8                            "\nCapacity: " + SamsungT5.
   getCapacity() + "TB" +
 9                            "\nTeraBytes Written (TBW): "
    + SamsungT5.getTeraBytesWritten()
10                            );
11     }
12 }
13 
```

```java

public class Assignment3b {
    public static void main(String[] args) {

        Fruit banana = new Fruit("Banana", "Yellow");
        Fruit rambutan = new Fruit("Rambutan", "Red");

        Fruit otherBanana = new Fruit(banana);
        Fruit otherRambutan = new Fruit(rambutan);


        //if banana is the same fruit as the other banana, then they are the same fruit
        if (banana.equals(otherBanana)) {
            System.out.println(banana.toString() + " is equal to " + otherBanana.toString());
            System.out.println("They are the same fruit!");
        //otherwise, they are not the same fruit
        } else {
            System.out.println("They are NOT the same fruit!");
        }
    }
}

```

```java

public class Assignment3c {
    public static void main(String[] args) {
        Zombie zombie = new Zombie("Rotting Ron");
        Hero hero = new Hero("Heroic Hector");

        //while the zombie and the hero are both alive, the game will continue running
        while (zombie.isAlive() && hero.isAlive()) {
            //if the zombie has a higher roll, the zombie bites the hero
            if(zombie.roll() > hero.roll()) {
                zombie.bites(hero);
            //otherwise, the hero slashes the zombie
            } else {
                hero.slashes(zombie);
            }
        }
    }
}
```