```
import java.util.*;
/************************************************
 * This class puts together specific values
 * to build the game.
 *
 * @author Ben Parsell
 * @version 1.0.0 (11/20/2015)
 ***********************************************/
public class Game
{
    /** Arraylist for player inventory **/
    private ArrayList<Item> playerInventory;

    /** Player's last location **/
    private Room lastLocation;

    /** Player's current location **/
    private Room currentLocation;

    /** The Game's current message **/
    private String currentMessage;

    /** Did you win? **/
    private boolean win;

    /** Did you lose? **/
    private boolean lose;

    /** Is it time to search a body? **/
    private boolean timeToSearch;

    /** Gun ammo **/
    private int ammoCount;

    /** Item variables **/
    private Item M1_Garand;
    private Item grenade;
    private Item helmet;
    private Item pills;
    private Item beer;
    private Item ammo;
    private Item apple;
    private Item MRE;
    private Item germanSoldier;
    private Item mine;
    private Item germanGeneral;
    private Item tank;
    private Item radio;

    /** Room Variables **/
    private Room plane;
    private Room secondStoryFarmHouse;
    private Room firstStoryFarmHouse;
    private Room backyard;
    private Room openField;
    private Room cityEntrance;
    private Room drugStore;
    private Room apartments;
    private Room tavern;
    private Room germanCamp;
    private Room generalsTent;
    private Room hillTop;

    /*******************************************************
     * Default Constructor
     *******************************************************/
    public Game() {
        // Player starts with gun
        playerInventory = new ArrayList<Item>();

        // Creates rooms
        createRooms();

        // Instantiate lastlocation
        lastLocation = null;

        // Instantiate currentLocation as plane
        currentLocation = plane;

        ammoCount = 4;
        // FIX ME (used for later)
        win = false;
        lose = false;

    }


    /*******************************************************
     * Method to retrieve currentLocation
     *
     * @return currentLocation player's location
     *******************************************************/
    public Room getCurrentLocation() {
        return currentLocation;
    }


    /*******************************************************
```

```
     * Method to set the initial message for player
     ****************************************************/
    public void setWelcomeMessage() {
        currentMessage = "Welcome, soldier! You are a WWII 101st Airborne paratrooper. You are dropping over northern France \n to help in the D-Day operation. You've noticed that the drop planes have overshot their destination though... \n";
    }


    /*****************************************************
     * Method to check playerInventory for a specific
     * item
     *
     * @return sub temp variable for item name
     ****************************************************/
    private Item checkForItem(String name) {
        Item sub = null;
        for(Item itm: playerInventory){
            if(itm.getName().equals(name)){
                sub = itm;
            }
        }
        return sub;
     }


    /*****************************************************
     * Method to update currentMessage with list of
     * player inventory items
     ****************************************************/
    public void list(){
        currentMessage = "";
        for(Item itm: playerInventory){
            currentMessage = currentMessage + itm.getName() + " " + "\n";
        }
    }


    /*****************************************************
     * Method to create rooms and items
     ****************************************************/
    private void createRooms() {
        // Create Items
        M1_Garand = new Item("M1 Garand", "a beat up semi-auto rifle that holds 8 rounds", 5, false);
        grenade = new Item("Grenade", "an unused pineapple-style hand grenade", 1, false);
        beer = new Item("Beer", "a bottle of warm beer", 3, true);
        pills = new Item("Pills", "a bottle of pills with the prescription label ripped off", 1, true);
        ammo = new Item("M1 Garand Ammo", "an 8 round ammo clip for the M1 Garand Rifle, your gun is reloaded.", 1, false);
        MRE = new Item("MRE", "a gross, meal that consists of canned corn, cold pulled-pork, and candy", 1, true);
        apple = new Item("Apple", "a fresh, Macintosh apple", 1, true);
        germanSoldier = new Item("German Soldier", "a combat-ready soldier trying to kill you.", 200, false);
        mine = new Item("Mine", "a primed, exlposive device ready to go off if stepped on.", 250, false);
        germanGeneral = new Item("German General", "a german general shining with medals of his acomplishments", 210, false);
        tank = new Item("Panzer Tank", "a Panzer Tiger tank loaded and ready to fire.", 3340, false);
        radio = new Item("Radio", "a radio to call for reinforcements", 20, false);

        // Create Rooms
        plane = new Room("the airplane. It is taking heavy AA fire, you parachute out, and have to select a location to land", M1_Garand);
        secondStoryFarmHouse = new Room("a destoryed-interior upstairs to a house. There is a dead \nAmerican paratrooper on the floor", germanSoldier);
        firstStoryFarmHouse = new Room("You are in the main living room of the farmhouse. There are exits at the front and back-door,\n everything else is blocked off.", germanSoldier );
        backyard = new Room("A fenced-in backyard with a horse stable.", germanSoldier);
        openField = new Room("An open field filled with mines. They can safely be blown up...", mine);
        cityEntrance = new Room("The gate to Main Street.", null);
        drugStore = new Room("a traditional 1940s drug store with a soda bar.", pills);
        apartments = new Room("beat up apartments with bullet holes in the walls.", germanSoldier);
        tavern = new Room("a French-style tavern. It's very dark in here...", germanSoldier);
        germanCamp = new Room("the hefty german camp.", tank);
        generalsTent = new Room("a shaggy military tent that has a radio in it.", germanGeneral);
        hillTop = new Room("a hilltop that has radio reception.", null);

        // adding neighbors
        plane.addNeighbor("north", secondStoryFarmHouse);
        plane.addNeighbor("south", openField);

        // Farm House 2nd Story
        secondStoryFarmHouse.addNeighbor("downstairs" , firstStoryFarmHou
se);

        // Farm Hosue 1st Story
        firstStoryFarmHouse.addNeighbor("north", backyard);
        firstStoryFarmHouse.addNeighbor("upstairs", secondStoryFarmHouse);
        firstStoryFarmHouse.addNeighbor("south", cityEntrance);

        // Open Field
        openField.addNeighbor("west", cityEntrance);

        // backyard
        backyard.addNeighbor("south", firstStoryFarmHouse);

        // City / Gate Entrance
        cityEntrance.addNeighbor("east", openField);
        cityEntrance.addNeighbor("south", drugStore);
        cityEntrance.addNeighbor("north", firstStoryFarmHouse);

        // Drugstore
        drugStore.addNeighbor("north", cityEntrance);
        drugStore.addNeighbor("east", tavern);
        drugStore.addNeighbor("upstairs", apartments);

        // Apartments
        apartments.addNeighbor("downstairs", drugStore);

        // Tavern
        tavern.addNeighbor("west", drugStore);
        tavern.addNeighbor("south", germanCamp);

        // German Camp (Boss)
        germanCamp.addNeighbor("north", tavern);
        germanCamp.addNeighbor("east", generalsTent);

        generalsTent.addNeighbor("north", hillTop);

    }


    /*******************************************************
     * Method to determine when the game is over
     ******************************************************/
    public boolean gameOver() {
        // Failed to kill upstairs enemy
        if((currentLocation == firstStoryFarmHouse) && lastLocation == secondStoryFarmHouse && lastLocation.getItem() == germanSoldier) {
            currentMessage = "You failed to kill the enemy, and he shot you in the chest dropping you dead.";
            lose = true;
            return true;
        }

        // Failed to kill mainfloor enemy
        if((currentLocation == backyard) && (lastLocation == firstStoryFarmHouse) && (lastLocation.getItem() == germanSoldier)) {
            currentMessage = "You failed to kill the enemy, and he shot you in the chest dropping you dead.";
            lose = true;
            return true;
        }

        // Failed to blow up mine
        if((currentLocation == cityEntrance) && (lastLocation.getItem() == mine) && lastLocation == openField) {
            currentMessage = "You failed to destroy the mine, and it threw your body into the air, shredding your limbs.";
            lose = true;
            return true;
        }

        // Failed to kill
        if((currentLocation == backyard) && (lastLocation.getItem() == mine) && lastLocation == openField) {
            currentMessage = "You failed to destroy the mine, and it threw your body into the air, shredding your limbs.";
            lose = true;
            return true;
        }

        // Failed to kill apartments enemy
        if(currentLocation == drugStore && lastLocation == apartments && lastLocation.getItem() == germanSoldier) {
            currentMessage = "You failed to kill the enemy, and he shot you in the chest dropping you dead.";
            lose = true;
            return true;
        }

        // Failed to kill first floor enemy
        if((currentLocation == cityEntrance) && (lastLocation == firstStoryFarmHouse) && (lastLocation.getItem() == germanSoldier)) {
            currentMessage = "You failed to kill the enemy, and he shot you in the chest dropping you dead.";
            lose = true;
            return true;
        }

```

```java
        // Failed to kill backyard enemy
        if(currentLocation == firstStoryFarmHouse && lastLocation == backyard && lastLocation.getItem() == germanSoldier) {
            currentMessage = "You failed to kill the enemy, and he shot you in the chest dropping you dead.";
            lose = true;
            return true;
        }

        // Failed to kill tavern enemy
        if(currentLocation == germanCamp && lastLocation == tavern && lastLocation.getItem() == germanSoldier) {
            currentMessage = "You failed to kill the enemy, and he shot you in the chest dropping you dead.";
            lose = true;
            return true;
        }

        // Failed to kill tavern enemy
        if(currentLocation == drugStore && lastLocation == tavern && lastLocation.getItem() == germanSoldier) {
            currentMessage = "You failed to kill the enemy, and he shot you in the chest dropping you dead.";
            lose = true;
            return true;
        }

        // Ate posion pills and died
        if(currentMessage.equals("You have eaten pills.\n")) {
            currentMessage = "You thought they were tasty... but they poisioned you! Game over!";
            lose = true;
            return true;
        }

        // This is to remove ammo from inventory. Has no impact on gameOver, but it would
        // have been uneccessary to add a entire new method for just this.
        if(lastLocation == backyard && checkForItem("M1 Garand Ammo") != null) {
            playerInventory.remove(ammo);
            return false;
        }


        // Successfully beat game
        if(currentLocation == hillTop && checkForItem("Radio") == radio && currentLocation.getItem() == null) {
            currentMessage = "You called for reinforcements and successfu
lly helped with the D-Day operation!";
            win = true;
            return true;
        }
        // Otherwise, game continues
        else {
            return false;
        }
    }


    /*******************************************************
     * Method to move the player between rooms
     ******************************************************/
    public void move(String direction) {
        // Create new room as you move
        Room next = currentLocation.getNeighbor(direction);

        // Error catch and actual movement
        if( next == null) {
            currentMessage = "You can't go that way" + "\n";
        } else {
            lastLocation = currentLocation;
            currentLocation = next;
            currentMessage = currentLocation.getLongDescription();
        }
    }


    /*******************************************************
     * Method to search body/continue on with game
     ******************************************************/
    public void continueGame() {
        // Searches dead enemy on secondFloor
        if(currentLocation == secondStoryFarmHouse) {
            currentLocation.removeItem();
            currentLocation.addItem(grenade);
            currentMessage = "You search the enemy body and see something
. Pickup " + currentLocation.getItem().getName() + "?\n";
            timeToSearch = false;
        }

        // Searches dead enemy in living room
        if(currentLocation == backyard) {
            currentLocation.removeItem();
            currentLocation.addItem(ammo);
            currentMessage = "You search the enemy body and see something
. Pickup " + currentLocation.getItem().getName() + "?\n";
```

```
            ammoCount = 8;
            timeToSearch = false;
        }

        // Searches dead enemy in backyard
        if(currentLocation == firstStoryFarmHouse) {
            currentLocation.removeItem();
            currentLocation.addItem(apple);
            currentMessage = "You search the enemy body and see something. Pickup " + currentLocation.getItem().getName() + "?\n";
            timeToSearch = false;
        }

        // Searches minefield
        if(currentLocation == openField) {
            currentLocation.removeItem();
            currentLocation.addItem(apple);
            currentMessage = "You search the now safe minefield and see something. Pickup " + currentLocation.getItem().getName() + "?\n";
            timeToSearch = false;
        }

        // Searches Tavern
        if(currentLocation == tavern) {
            currentLocation.removeItem();
            currentLocation.addItem(beer);
            currentMessage = "You search the enemy body and see something. Pickup " + currentLocation.getItem().getName() + "?\n";
            timeToSearch = false;
        }

        // Removes german from apartments
        if(currentLocation == apartments) {
            currentLocation.removeItem();
            currentMessage = "You may move on.\n";
            timeToSearch = false;
        }

        // Removes tank from camp
        if(currentLocation == germanCamp) {
            currentLocation.removeItem();
            currentMessage = "You may now move on. \n ";
            timeToSearch = false;
        }

        // Searches dead general
        if(currentLocation == generalsTent) {
            currentLocation.removeItem();
            currentLocation.addItem(radio);
            currentMessage = "You search the enemy body and see something. Pickup " + currentLocation.getItem().getName() + "?\n";
            timeToSearch = false;
        }
    }


    /****************************************************
     * Method to fire your weapon
     ***************************************************/
    public void fire() {
        // Check for rifle
        if(checkForItem("M1 Garand") == null) {
            currentMessage = "You do not have a rifle.\n";
        }

        // Check for ammo in rifle
        else if(ammoCount < 1) {
            currentMessage = "You are out of ammo.\n";

        // Check for the general
        }else if(currentLocation.getItem() == germanGeneral && checkForItem("M1 Garand") == M1_Garand && ammoCount > 0) {
            currentMessage = "You killed the General! \n";
            timeToSearch = true;
            ammoCount -= 1;
        }

        // Check for a german soldier
        else if(currentLocation.getItem() == germanSoldier && checkForItem("M1 Garand") == M1_Garand && ammoCount > 0) {
            currentMessage = "You killed the enemy. CLICK CONTINUE \n";
            timeToSearch = true;
            ammoCount -= 1;

        // Check for the mine
        } else if(currentLocation.getItem() == mine && checkForItem("M1 Garand") == M1_Garand && ammoCount > 0) {
            currentMessage = "You shoot the mine, and a huge explosion of dirt fills the sky. You may now move forward. CLICK CONTINUE\n";
            timeToSearch = true;
            ammoCount -= 1;

        // Check for the tank
        } else if(currentLocation.getItem() == tank && checkForItem("Grenade") == grenade) {
            currentMessage = "You ran up to the tank, jumped on it, and threw the grenade into the hatch killing the crew. CLICK CONTINUE\n";
            playerInventory.remove("Grenade");
            timeToSearch = true;
        }

        // Otherwise, you cannot fire now
        else {
            currentMessage = "You don't need to fire now, there are no en
emies.\n";
        }
    }


    /*****************************************************
     * Custom method to check if it is time to search
     * an enemy body (works with buttons)
     ****************************************************/
    public boolean isTimeToSearch() {
        // Check to see if it is time to search
        if(timeToSearch) {
            return true;
        }
        return false;
    }


    /*****************************************************
     * Method to move the player between rooms
     ****************************************************/
    public void look() {
        // Will display message to GUI later
        currentMessage = currentLocation.getLongDescription() + "\n";
    }


    /*****************************************************
     * Method to retrieve current message
     *
     * @return currentMessage the message for the game
     ****************************************************/
    public String getMessage() {
        return currentMessage;
    }


    /*****************************************************
     * Method to display a help message to the user
     ****************************************************/
    public void help() {
        currentMessage = "You are an American paratropper behind enemy li
nes. You need to eliminate the German general before you can\n safely rec
```

```
onnect with your men. Look for the German camp... \n";
    }


    /*****************************************************
     * Method to pickup an item if there is one in
     * the currentRoom
     *****************************************************/
    public void pickup(String item) {
        // Check if room has an item at all
        if(currentLocation.hasItem() == false){
            currentMessage = "There is no item in the room to take" + "\n
";
        }

        // Check if the item is too heavy to lift
        else if(currentLocation.getItem().getWeight() > 100){
            currentMessage = "That item is too heavy to pickup" + "\n";
        }

        // Otherwise, pickup the item
        else if(item.equals(currentLocation.getItem().getName())){
            playerInventory.add(currentLocation.getItem());
            currentMessage = "You have picked up " + currentLocation.getI
tem().getName() + "\n";
            currentLocation.removeItem();
        }
    }


    /*****************************************************
     * Method to drop an item from player inventory
     *****************************************************/
    public void drop(String item) {
        Item roomItem = null;
        boolean haveItem = false;

        // Check to see if item already has an item
        if(currentLocation.hasItem() == true) {
            currentMessage = "Room has an item already.";
        }

        // Check to make sure inventory has an item
        if(playerInventory.size() == 0) {
            currentMessage = "You have nothing to drop.";
        }

        // Otherwise, run through inventory & drop
        else {
```

```java

            // Cycles through inventory
            for(Item i : playerInventory) {

                // Check to make sure names match
                if(i.getName().equals(item)) {
                    roomItem = i;
                    haveItem = true;
                    currentMessage = "You dropped an item.";
                }
            }

            // Remove item and add it to room
            playerInventory.remove(roomItem);
            currentLocation.addItem(roomItem);

            // Make sure you have an item
            if(haveItem == false) {
                currentMessage = "You aren't holding an item.";
            }
        }
    }


    /*******************************************************
     * Eat an item if it's edible
     ******************************************************/
    public void eat(String item) {
        Item canEat = null;

        // Find item in player inventory
        for(Item itm: playerInventory){
            // Check for matches and set temp variable equal
            if(itm.getName().equals(item)){
                canEat = itm;
            }
        }


        // Check to see if the item is in the inventory
        if (canEat == null){
            currentMessage = "You don't have that item"+ "\n";
        }

        // If it is, continue
        else{

            // Check to see if item is edible
            if(canEat.isEdible() == true && item.equals("Pills")) {
                currentMessage = "You have eaten pills.\n";
                playerInventory.remove(canEat);
            }
            if(canEat.isEdible() == true && !item.equals("Pills")){
                currentMessage = "Yum that was a tasty " + canEat.getName
() + "!" + "\n";
                playerInventory.remove(canEat);
            }

            // If it is not, update message
            else if(canEat.isEdible() == false){
                currentMessage = canEat.getName() + " is not edible." + "
\n";
            }
        }
    }
}

```